Titre: Title:	Intégrer une approche de conception centrée utilisateur à une approche agile de développement logiciel
Auteur: Author:	Jean-François Proulx
Date:	2010
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Proulx, J.-F. (2010). Intégrer une approche de conception centrée utilisateur à une approche agile de développement logiciel [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/455/

UNIVERSITÉ DE MONTRÉAL

INTÉGRER UNE APPROCHE DE CONCEPTION CENTRÉE UTILISATEUR À UNE APPROCHE AGILE DE DÉVELOPPEMENT LOGICIEL

JEAN-FRANÇOIS PROULX

DÉPARTEMENT DE MATHÉMATIQUES ET GÉNIE INDUSTRIEL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION

DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES

(GÉNIE INDUSTRIEL)

DÉCEMBRE 2010

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé:

INTÉGRER UNE APPROCHE DE CONCEPTION CENTRÉE UTILISATEUR À UNE APPROCHE AGILE DE DÉVELOPPEMENT LOGICIEL

présenté par : PROULX JEAN-FRANÇOIS

en vue de l'obtention du diplôme de : Maîtrise ès sciences appliquées, Génie industriel

a été dûment accepté par le jury d'examen constitué de :

M. PELLERIN, Robert, ing., Ph.D., président

M. ROBERT, Jean-Marc, Ph.D., membre et directeur de recherche

M. ROBILLARD, Pierre-N., ing., Ph.D., membre

DÉDICACE

À vous, les équipes Scrum, qui veulent réaliser du logiciel réellement utile,

À vous, les ergonomes, qui désirent contribuer à réaliser du logiciel fonctionnel,

Ce travail est pour nous tous, artisans du logiciel,

Unissons nos efforts et les utilisateurs de logiciels ne s'en porteront que mieux.

REMERCIEMENTS

Aux équipes Scrum que j'ai côtoyées, vos questions m'ont inspiré.

À Jean-Marc, tes conseils m'ont guidé.

À Karine, ta patience et tes encouragements m'ont motivé.

À mes chiens, les promenades de fin de soirée m'ont gardé éveillé.

Merci!

RÉSUMÉ

Ce travail traite de l'intégration de l'approche de conception centrée utilisateur (CCU) à une approche agile de développement logiciel, plus spécifiquement à l'approche Scrum. Il présente les éléments clés de chacune des approches et propose une façon pratique de les articuler ensemble pour que le résultat d'intégration soit facile à mettre en oeuvre par les équipes de développement logiciel. Nous faisons ressortir des similarités entre les approches qui favorisent une intégration des pratiques, ainsi que des défis envisagés qu'il est nécessaire de surmonter pour réussir cette intégration. Pour atteindre l'objectif d'intégrer les pratiques de chaque domaine, nous présentons dans un guide destiné aux équipes de développement logiciel, une méthode d'intégrer des activités de CCU à celles de l'approche Scrum. Le guide s'adresse d'abord à des équipes de développement logiciel suivant une approche Scrum, qui désirent intégrer des pratiques de conception centrée utilisateur.

Nos expériences à travailler selon une approche Scrum et à tenter d'y appliquer des pratiques de conception centrée utilisateur ont motivé ce travail. La réalisation des projets amenait des questions qui étaient laissées sans réponse satisfaisante par les travaux d'autres auteurs. Certains travaux n'apportaient pas d'éléments de réponse, mais plutôt un argumentaire pour l'intégration des approches. D'autres, qui tentent des pistes de solutions, ne semblent pas pleinement adhérer aux principes d'une approche agile. Malgré ceci, les différents travaux des auteurs apportent des pistes de réponses qui peuvent être mises en commun. Ce sont ces éléments qui ont contribué à concevoir le guide présenté.

Le guide proposé comprend 18 activités distinctes qui proviennent autant de l'approche de la CCU que de l'approche Scrum. Il tente de respecter les principes de chaque approche dans le choix des activités et dans la séquence appliquée durant les étapes de conception et de développement d'un logiciel.

L'évaluation du guide proposé se fait en deux phases : dans la première phase, cinq spécialistes de la CCU et six spécialistes du développement agile lisent le guide et proposent des

améliorations, s'il y a lieu. Ceci permet de vérifier si les activités présentées respectent les principes de chaque approche et semblent contribuer à la réalisation d'un logiciel utile pour les utilisateurs. Le travail présente les deux phases d'évaluation, mais discute seulement des résultats obtenus durant la première phase d'évaluation. Dans la deuxième phase, des équipes agiles ayant reçu le guide seront suivies pour connaître l'applicabilité du guide dans un contexte réel de projet et l'impact sur le travail des différents membres des équipes. Des ajustements seront apportés au guide à l'issue de chacune des étapes de validation, si nécessaire.

Ce travail de conception et d'évaluation d'une méthode pour intégrer des activités de CCU à celles de l'approche Scrum a été présenté sous forme d'un article court lors du colloque IHM 2010 – 22ème conférence francophone sur les interactions Homme-machine [1].

ABSTRACT

This work deals with the integration of a user-centered design (UCD) approach to an agile software development approach, specifically the Scrum approach. It starts by describing the key elements of each approach and offers a convenient way to link them together so that the result of integration is easy to implement by software development teams. We highlight similarities between the approaches that promote integration of practices and the challenges to overcome to succeed in this integration. To meet this integration goal, we describe a method to integrate UCD activities with those of the Scrum approach in a guide presented to software development teams. The guide is primarily intended for software development teams using a Scrum approach that wish to integrate user-centered design practices.

Our experience working with a Scrum approach and trying to apply user-centered design practices have motivated this work. Projects brought questions that were left without a satisfactory response by the work of other authors. Some work did not lead to answers, but rather brought an argument for the integration of approaches. Others who were trying possible solutions did not to fully endorse the principles of an agile approach. Despite this, the work of different authors contributed some clues that can be brought together. These are the elements that contributed to the design of this guide.

The proposed guide includes 18 separate activities. These activities come from both a UCD approach and the Scrum approach. The guide tries to respect the principles of each approach in the choice of activities and sequence of steps applied during the design and development software.

Finally, the evaluation of the proposed guide is done in two stages: in the first phase, five UCD experts and six agile development specialists read the guide and suggest improvements, if any. This verifies whether the proposed activities meet the principles of each approach and appear to contribute to the achievement of useful software for users. This work presents the two evaluation phases, but only discusses the results obtained during the first evaluation phase. In the second

stage, we will follow agile teams who received the guide to determine the applicability of the guide in a real project and the impact on the work of different team members. Adjustments will be made to the guide at the end of each stage of validation, if necessary.

This design and evaluation work for a method of integrating UCD activities into those of the Scrum approach was presented in the form of a short article at the IHM 2010 conference – 22ème conférence francophone sur les interactions Homme-machine [1].

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

CCU	Conception centrée utilisateur
IHM	Interface humain-machine
PP	Propriétaire du produit (*product owner*)

LISTE DES ANNEXES

INTRODUCTION

Usability is one of those things that is first understood in the negative. By that I mean, it is often easier to know when something isn't usable than when it is [2].

Everyone will like Scrum; it is what we already do when our back is against the wall [3].

Les approches agiles de développement logiciel gagnent en popularité puisqu'elles réussissent à livrer fréquemment du logiciel de valeur qui répond aux besoins et aux attentes du client. En parallèle, les utilisateurs de systèmes informatiques demandent de plus en plus que les logiciels soient adaptés à leurs besoins et leurs attentes, ce qui nécessite une approche ergonomique au développement logiciel. L'intégration des approches d'ergonomie logicielle et agile semble nécessaire puisqu'elle permettrait, en principe, d'augmenter la valeur du logiciel développé autant pour le client que pour l'utilisateur final. Pourtant il existe encore un fossé entre les communautés de chaque domaine. Les auteurs praticiens et théoriciens apportent leurs expériences respectives pour rapprocher les communautés; alors le fossé se rétrécit, mais il est encore présent.

La problématique qui émane de la différence entre les philosophies des deux approches est vécue quotidiennement par l'auteur. Nous tentons d'appliquer les pratiques du domaine de l'ergonomie logicielle, qui demandent généralement de faire beaucoup de travail en amont du projet de développement logiciel, dans un environnement agile où on préconise de faire de la conception pour répondre aux besoins en cours et au mode juste à temps.

Mise en situation

Depuis quelques années, un certain nombre d'auteurs des domaines de l'ergonomie et de l'agilité (Jeff Patton, Don Norman, Scott Ambler, Jakob Nielsen, Larry Constantine, etc.) s'intéressent à l'intégration de l'ergonomie et des approches agiles, plus spécifiquement Scrum. Ces auteurs

traitent généralement de retour d'expérience, proposent des approches utilisant des pratiques de chaque domaine, avancent des raisons ou des avantages de jumeler les approches. Cette littérature est utile pour bâtir un argumentaire en faveur de l'intégration mais ne répond pas nécessairement aux questions qu'ont les praticiens lorsqu'ils tentent de faire cette intégration. Par exemple, en adoptant en amont du projet les pratiques de conception centrée sur l'utilisateur, est-ce que les principes agiles peuvent être respectés? Est-il utile et efficace d'effectuer des tests d'utilisabilité pour obtenir une rétroaction des utilisateurs sans nuire au déroulement d'une itération de développement du logiciel? Jusqu'à quel point les activités d'ergonomie telles que l'analyse de tâche et du contexte, la définition des caractéristiques des utilisateurs, la participation active des utilisateurs dans le processus de conception, la mesure de performance des utilisateurs sont-elles nécessaires et doivent être poussées pour obtenir un logiciel de qualité qui réponde aux besoins et attentes des utilisateurs? Les réponses à ces questions ne sont pas encore toutes disponibles, ainsi beaucoup essaient d'apprendre des succès des autres [4].

L'intégration est généralement traitée d'un point de vue d'ergonomes agissant dans des projets de développement logiciel et désirant mettre en place une approche agile, et très peu du point de vue d'agilistes qui tentent d'appliquer des pratiques d'ergonomie. Pourtant, quoique la motivation soit la même : développer du meilleur logiciel qui réponde aux besoins des utilisateurs, la vision est différente. Ainsi, il est nécessaire de répondre aux questions précédentes en considérant les points de vue de chaque communauté.

Objectif

Ce mémoire a pour objectif de guider des équipes travaillant selon un mode agile/Scrum à intégrer des pratiques d'ergonomie. Les propositions que nous faisons tenteront de répondre à certaines questions laissées sans réponses satisfaisantes par les auteurs qui s'intéressent à l'intégration des pratiques d'ergonomie et des pratiques agiles de développement logiciel. Les propositions seront présentées sous forme d'un guide pratique combinant différentes approches et ajoutant des conseils d'utilisation. Le guide proposé est basé sur des expériences acquises par

l'auteur de ce mémoire à travers différents projets de développement logiciel au sein de l'entreprise Pyxis Technologies.

Structure du document

Le chapitre 1 présente d'abord l'approche agile/Scrum et l'approche ergonomique de développement logiciel. Le chapitre 2 présente la situation actuelle de l'intégration de l'ergonomie logicielle à l'approche agile, puis les défis rencontrés ou envisagés lors de cette intégration. Le chapitre 3 propose les principes menant à la conception du guide pratique, qui aidera à effectuer les activités d'ergonomie dans un cadre agile. Ce guide tente de combler certaines lacunes dans les travaux des auteurs qui ont étudié le sujet. Le chapitre 4 présente une évaluation du guide proposé et des recommandations pour les équipes de développement logiciel qui désirent intégrer les pratiques des deux approches. Dans la conclusion, nous proposons quelques avenues de recherche qui semblent pertinentes pour faire avancer les connaissances sur le sujet.

Chapitre 1 APPROCHES DE DÉVELOPPEMENT LOGICIEL

Ce chapitre présente les principes qui guident et les pratiques qui traduisent les approches agiles et d'ergonomie dans le développement logiciel. Nous présentons d'abord les valeurs et les principes qui gouvernent les approches agiles et la méthode Scrum plus particulièrement. Ensuite, l'approche ergonomique de développement logiciel est abordée en décrivant les principaux thèmes qui forment cette approche.

1.1 LES APPROCHES AGILES

Les approches agiles sont un ensemble de pratiques de développement itératif et incrémental de logiciel, misant sur des courts cycles de développement, afin de concevoir rapidement des solutions informatiques ayant de la valeur pour le client. Ambler [5] précise qu'il est difficile de trouver une définition spécifique de ce qu'est le développement logiciel agile. Il propose une définition intéressante contenant les éléments principaux qui aident à acquérir une compréhension globale de l'approche;

> Un processus itératif et incrémental de développement logiciel
> exécuté dans un esprit de collaboration
> par des équipes auto organisées, dans un cadre de gouvernance efficace
> avec suffisamment de cérémonie
> qui produit des logiciels de haute qualité
> dans un bon rapport coût-efficacité et en temps voulu
> qui répond aux besoins changeants de ses parties prenantes.
>
> [3, traduction libre de l'auteur, formatage de l'auteur original]

Cette définition fait ressortir, entre autres, les quatre valeurs fondamentales que prônent les approches agiles : l'équipe, le logiciel fonctionnel, la collaboration et l'acceptation du changement. Ces valeurs sont d'ailleurs à la base du Manifeste agile [6] et des principes [7] qui en découlent. Le manifeste a été rédigé par 17 experts du domaine du développement logiciel qui estimaient que les approches traditionnelles n'étaient pas efficaces pour concevoir des logiciels de qualité qui répondent aux besoins des clients. Les approches traditionnelles sont plus axées sur la

planification et le contrôle des échéanciers, des dates cibles, du respect des budgets et de la documentation initiale.

1.1.1 Manifeste agile

Les approches agiles ne sont pas nées suite à la rédaction du Manifeste. Par contre, ce dernier est une formalisation de ces approches qui deviendront « agiles ». L'objectif du Manifeste est de promouvoir une approche différente de développement logiciel en s'attachant à délivrer, de la meilleure façon possible, ce qui a de la valeur pour le client [8]. Selon la terminologie agile, le client n'est pas une personne externe au projet qui achète un logiciel, mais plutôt une personne qui remplit un rôle tout au long du déroulement du projet. Le Manifeste agile prône quatre valeurs et est considéré comme l'acte généralisateur des approches agiles [9].

Les quatre valeurs agiles [5, traduction libre de l'auteur] sont les suivantes :

- les individus et les interactions priment plutôt que les processus et les outils;
- les logiciels fonctionnels priment plutôt qu'une documentation exhaustive;
- la collaboration avec le client prime plutôt que la négociation de contrat;
- la réponse au changement prime plutôt que le suivi d'un plan.

Le Manifeste révèle que, pour chaque valeur, les éléments de gauche ont plus de valeur que ceux de droite, mais on reconnaît l'utilité des éléments de droite. Les outils et les processus, la documentation et un plan demeurent utiles et nécessaires. Cependant, dans les approches agiles les éléments de gauche ne doivent pas être mis de l'avant au détriment des autres qui sont abordés dans chacune des valeurs. Les valeurs sont simples et faciles à mémoriser. Cependant, elles méritent d'être détaillées pour guider et faciliter la mise en place d'une approche agile. Les valeurs sont supportées par 12 principes.

1.1.2 Douze principes agiles

Les principes agiles apportent des éléments plus concrets aux valeurs du Manifeste. Ainsi, ils cadrent plus avec les comportements des équipes agiles en énonçant des méthodes de travail à employer, des résultats envisagés ou des objectifs à viser. Les 12 principes agiles [6, traduction libre de l'auteur] sont les suivants :

- Notre plus haute priorité est de satisfaire le client en lui livrant rapidement, et ce, de façon continue, un logiciel de qualité.
- Accepter les changements de besoins, même lors du développement. Les processus agiles exploitent les changements pour augmenter les avantages compétitifs du client.
- Livrer fréquemment un logiciel fonctionnel en visant les délais les plus courts, de quelques semaines à quelques mois.
- Gestionnaires et développeurs doivent travailler ensemble, de façon quotidienne, pour toute la durée du projet.
- Bâtir des projets autour d'individus motivés. Donnez-leur l'environnement et le support dont ils ont besoin, et ayez confiance qu'ils feront le travail.
- La méthode la plus efficace pour transmettre l'information à l'équipe de développement et à l'intérieur de celle-ci est par la conversation de personne à personne.
- Un logiciel fonctionnel est la mesure principale de l'avancement.
- Les processus agiles favorisent le développement durable. Les responsables, développeurs et utilisateurs devraient pouvoir conserver un rythme constant indéfiniment.
- Une attention continue à l'excellence technique et une bonne conception augmentent l'agilité.
- La simplicité – l'art de minimiser la quantité de travail fait inutilement – est essentielle.
- Les meilleures architectures, exigences et conceptions surgissent d'équipes autoorganisées.
- À intervalles réguliers, l'équipe réfléchit à une façon de devenir plus efficace, puis adapte et ajuste son comportement en conséquence.

C’est en considérant et appliquant les 12 principes qu’une approche agile peut réellement être matérialisée.

1.1.3 Scrum

Scrum est une des principales approches agiles [10][11] de gestion et de suivi d’avancement de projets et *eXtreme Programming* (*XP*) [12] est une des méthodes les plus populaires pour guider la programmation et les pratiques techniques dans le développement logiciel.

Le terme Scrum provient de la mêlée au rugby pour symboliser l’équipe qui travaille de manière auto organisée et qui s'adapte selon la situation de jeu. L'équipe fait avancer le jeu de façon efficace et efficiente sans avoir prédéfini un plan de jeu complet [13], où les membres de l'équipe connaissent bien l'objectif commun d'équipe, qui est de marquer un point.

Les rituels de Scrum font qu’autant les développeurs que le client sont nécessaires à l’atteinte de l’objectif d’équipe. Un représentant du client nommé propriétaire du produit (*product owner*) est responsable de transmettre à l’équipe l’orientation du projet, de définir les fonctionnalités et de suggérer l’ordre dans lequel elles devraient être développées par l’équipe afin de fournir un logiciel qui a de la valeur pour lui. Il exécute ceci dans le carnet de produit (*product backlog*). Le carnet de produit est une liste d'éléments, représentant des besoins ou fonctionnalités désirées par le propriétaire du produit. Les éléments y sont classés selon la valeur d'affaire accordée par le propriétaire du produit. La valeur accordée aux éléments par le propriétaire dépend des critères qu'il considère : le retour sur investissement (*ROI*), la criticité d'une fonctionnalité dans le système ou pour les utilisateurs, le coût en effort de développer la fonctionnalité, etc. Le carnet de produit est visible pour toute l’équipe et permet ainsi une meilleure communication puisque tous les membres de l’équipe peuvent voir les fonctionnalités demandées par le propriétaire du produit.

Le travail de développement est découpé en itérations (*sprints*) qui ont généralement une durée de trois ou quatre semaines. Cette courte durée permet de livrer rapidement au propriétaire du produit un incrément de logiciel comportant le plus de valeur d'affaire. Le logiciel livré a de la valeur pour le propriétaire du produit puisque les fonctionnalités développées sont celles qui ont la plus haute priorité, celles se trouvant en tête du carnet de produit. Un autre avantage de la courte durée des itérations est de permettre au propriétaire du produit de modifier les priorités des fonctionnalités demandées au fur et à mesure de l'avancement du projet de développement, et ce, selon l'évolution de sa réalité d'affaire. En effet, s'il désire ajouter des fonctionnalités, celles-ci seront potentiellement prises en compte lors de la prochaine itération. La figure 1 présente la flexibilité offerte au propriétaire du produit pour manipuler le contenu du carnet de produit durant le projet. À tout moment, il peut modifier l'ordre des éléments du carnet en considérant qu'un de ces éléments a plus de valeur qu'un autre, il peut insérer des éléments pour répondre à de nouveaux besoins ou en retirer si des besoins initialement mentionnés ne sont plus désirés. Une exception est posée au propriétaire du produit. Elle demande de ne pas modifier des éléments qui sont dans le carnet d'itération puisqu'ils sont en cours de réalisation par l'équipe de développement.

L'approche Scrum est considérée comme étant itérative et incrémentale. Elle est itérative puisque le développement du logiciel se produit par itérations et non dans une seule longue séquence. Ces cycles de courtes itérations offrent aux équipes la possibilité d'inspecter leur fonctionnement, de faire ressortir les points à améliorer et ceux qui sont efficaces. Ceci afin de perfectionner leurs pratiques de développement et de conserver les éléments qui permettent à l'équipe de bien opérer. L'approche incrémentale signifie, de son côté, que chaque élément développé durant une itération bâtit sur ce qui existe, tel que présenté dans la figure 2. Une fonctionnalité est développée pour répondre au besoin minimal exprimé par le propriétaire du produit. D'autres fonctions s'y grefferont pour augmenter l'ensemble fonctionnel et ainsi permettre au logiciel d'offrir un plus grand nombre de fonctionnalités utiles pour le propriétaire du produit. L'avantage est que chaque incrément offre une valeur supplémentaire au logiciel pour le client et celui-ci peut en prendre connaissance dès la fin de l'itération.

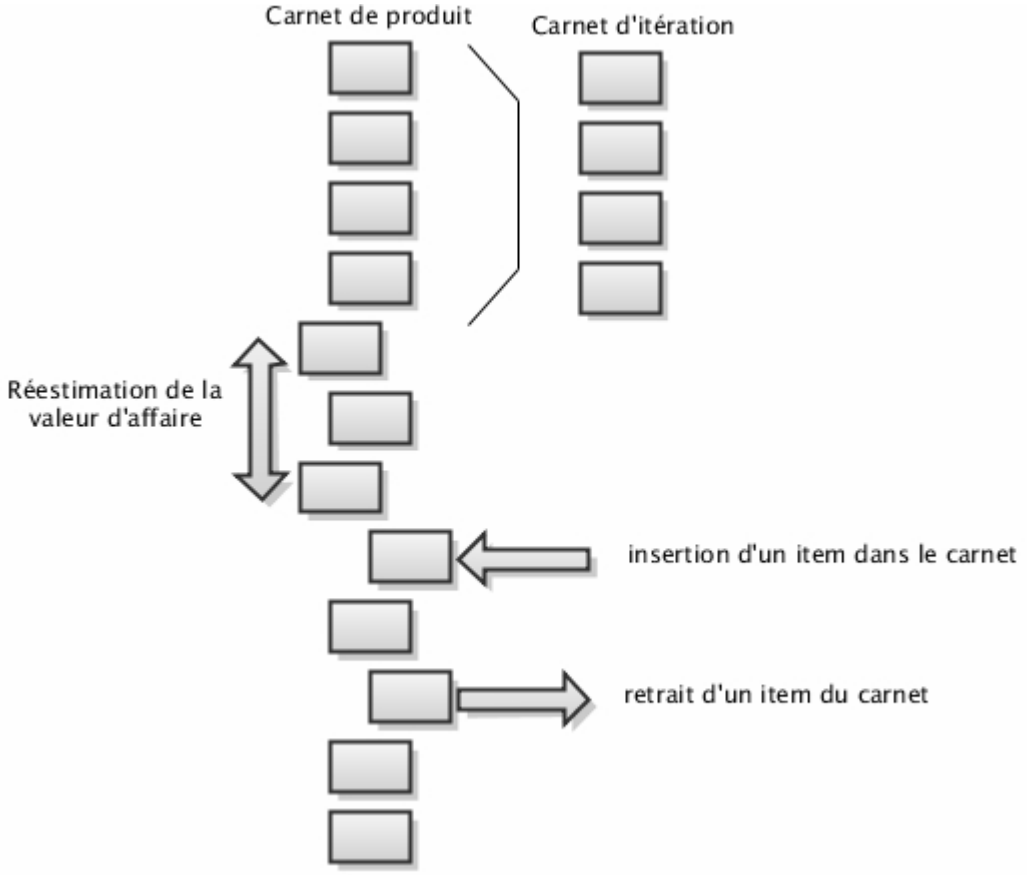


Figure 1 : Manipulation des éléments du carnet de produit.

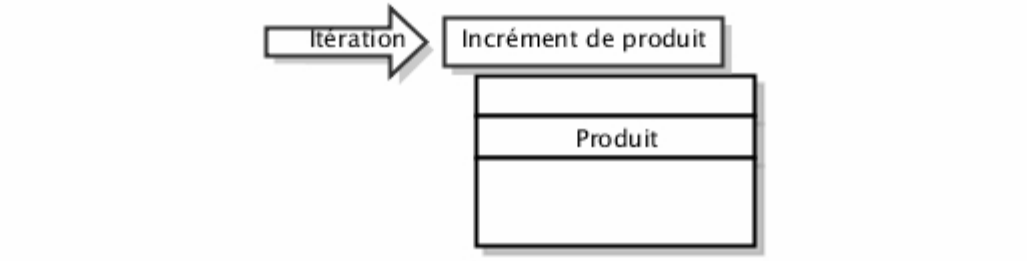


Figure 2 : Incrément de produit fonctionnel réalisé durant une itération.

En soi, Scrum ne suggère que trois cérémonies : la planification d'itération (*sprint planning*), la mêlée quotidienne (*daily Scrum*) et la revue d'itération (*sprint review*) [14]. Durant ces cérémonies, il y a production de trois artéfacts « de gestion » : le carnet de produit priorisé

(*product backlog*), le carnet d'itération (*sprint backlog*) et le graphique d'avancement (*burndown chart*) [15]. La figure 3 schématise le déroulement d'une itération.

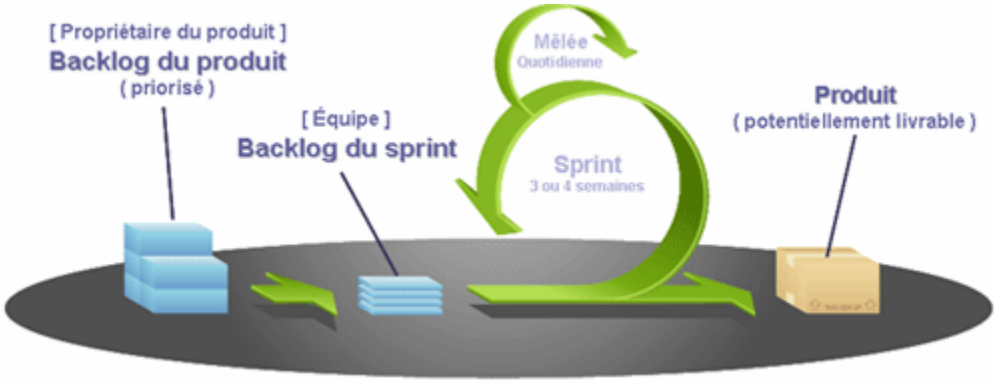


Figure 3 : Le déroulement d'une itération [16].

La planification d'itération est faite à partir du carnet de produit et a comme résultat le carnet d'itération. Durant l'itération, l'avancement fait par l'équipe est indiqué dans le graphique d'avancement et discuté durant la mêlée quotidienne. Lors de la revue d'itération, les éléments du carnet d'itération qui ont été complètement développés sont présentés au propriétaire du produit. Voyons maintenant plus en détails chaque activité et artéfact de Scrum.

Carnet de produit

Le carnet de produit présente une liste des besoins à satisfaire et des fonctionnalités désirées dans le système. Le propriétaire du produit est responsable et priorise les fonctionnalités selon leur valeur d'affaire respective. L'équipe de développement travaille en collaboration avec le propriétaire du produit afin de donner des estimations d'efforts de développement, basées sur sa compréhension des fonctionnalités énoncées. Cette estimation aide le propriétaire à ajuster la priorité associée aux besoins pour s'assurer que ceux qui sont les plus prioritaires permettent d'obtenir un meilleur retour sur investissement. Des nouvelles fonctionnalités peuvent être ajoutées ou retirées à tout moment du carnet de produit. Cette flexibilité offerte au propriétaire du produit est un grand avantage de Scrum.

Carnet d'itération

Le carnet d'itération est l'artéfact produit par l'activité de planification d'itération. Il présente les fonctionnalités que l'équipe s'engage à développer durant l'itération. Une équipe Scrum est une équipe auto organisée. C'est alors elle qui détermine le contenu de l'itération en fonction de sa capacité d'implémenter des fonctionnalités. Cet engagement de l'équipe est autant envers le propriétaire du produit qu'envers elle-même.

Graphique d'avancement

Le graphique d'avancement d'itération (figure 4) permet aux équipes de voir leur avancement par rapport au travail engagé dans l'itération en cours. L'axe vertical présente la quantité travail à faire et l'axe horizontal les jours de travail. Une courbe idéale d'avancement (trait pointillé) représente le travail projeté. Il indique le rythme de travail théorique que l'équipe devrait suivre pour respecter son engagement d'itération. La courbe idéale sert de référence pour l'équipe, afin de connaître si le rythme de travail réel (trait plein) est plus rapide (sous le trait pointillé) ou moins rapide que prévu (au dessus du trait pointillé). L'équipe, en collaboration avec le propriétaire du produit, peut alors réagir rapidement pour tenter de respecter l'engagement d'itération. De plus, ce graphique donne une visibilité quotidienne au propriétaire du produit sur l'avancement du travail de l'équipe.

Planification d'itération

Une planification d'itération débute avec un carnet de produit où sont identifiées suffisamment de fonctionnalités pour permettre le déroulement d'une itération. L'équipe de développement sélectionne les fonctionnalités, en débutant par le haut du carnet de produit, qu'elle développera durant l'itération qui débute et profite de la présence du propriétaire pour discuter de ce que chaque fonctionnalité comprend. La sélection se fait aussi en se basant sur la capacité de livraison de fonctionnalités de l'équipe, calculée à partir des fonctionnalités livrées dans les itérations passées.

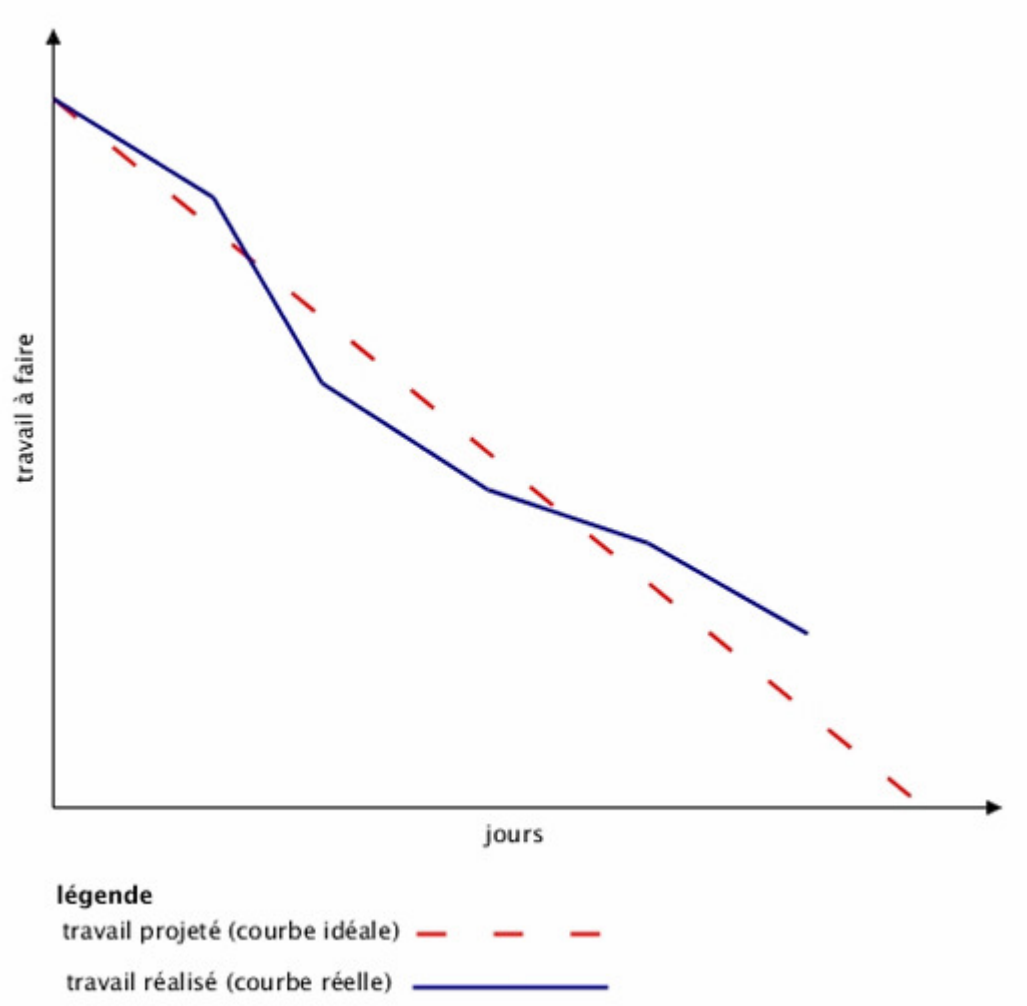


Figure 4 : Graphique d'avancement de l'itération

Mêlée quotidienne

La mêlée quotidienne est une rencontre de 15 minutes où l'équipe fait un compte rendu de son avancement vers l'atteinte de l'objectif de l'itération, où chaque membre de l'équipe répond à trois questions : qu'as-tu accompli depuis la dernière mêlée? Que vas-tu accomplir jusqu'à la prochaine mêlée? Est-ce que des éléments te bloquent dans ton avancement? Ces trois questions permettent à l'équipe de s'assurer qu'elle travaille bien à faire avancer l'itération vers l'objectif fixé et de découvrir des obstacles généraux ou des éléments qui étaient jusqu'alors inconnus et qu'il est nécessaire de prendre en compte dans le déroulement de l'itération. Cette rencontre est

quotidienne et permet à l'équipe de réagir rapidement aux obstacles et imprévus qui pourraient l'empêcher de remplir son engagement envers le propriétaire du produit.

Revue d'itération

La revue d'itération se fait à la fin du cycle de développement de l'itération et permet au propriétaire du produit de prendre connaissance des fonctionnalités qui ont été développées durant l'itération. À la revue d'itération, le propriétaire évalue si le produit qui lui est présenté correspond bien à l'engagement de livraison pris par l'équipe lors de la séance de planification de l'itération. Si le logiciel représente la valeur d'affaire initialement envisagée, il peut ensuite décider de cesser le développement et de livrer le produit pour une mise en production ou de poursuivre le développement pour ajouter les fonctionnalités restantes du carnet de produit. Après la revue d'itération, une nouvelle itération débute et relance ainsi le cycle de planification, développement et revue.

1.2 L'approche ergonomique dans le développement logiciel

L'ergonomie étudie les relations entre l'humain, sa tâche, son environnement et les outils, les machines ou systèmes utilisés pour faire sa tâche. Ainsi, l'ergonomie appliquée au développement logiciel (ergonomie logicielle) fait une étude plus spécifique des relations entre l'humain et l'environnement logiciel qu'il utilise. Pour arriver à cette fin, un ergonome fait appel à différentes approches et pratiques qui permettent de mettre en place les critères de conception et d'évaluation du logiciel développé. La conception centrée utilisateur (CCU) est une approche fréquemment utilisée qui permet de guider la conception d'interface humain-machine (IHM) en prenant en considération les caractéristiques des utilisateurs, et de focaliser sur l'utilisabilité du produit conçu. Les principaux thèmes de l'ergonomie logicielle seront maintenant décrits plus en détails.

1.2.1 Conception centrée utilisateur

La CCU consiste à tenir compte des caractéristiques des utilisateurs, des tâches qu'ils tentent d'accomplir et du contexte dans lequel ces dernières sont accomplies dans le développement d'artéfacts et à impliquer activement les utilisateurs tout au long du processus de développement d'une application informatique [17]. Il existe plusieurs normes guidant la planification et les activités de conception et de développement de systèmes qui répondent aux besoins et aux attentes de l'opérateur humain ou de l'utilisateur d'un logiciel; ISO9241 [18] est la principale norme dans ce domaine. Elle est découpée en plusieurs parties et elle présente, entre autres, six principes qui fournissent un cadre pour la conception centrée utilisateur, mais ne décrit pas de processus de conception spécifique. Ces principes sont les suivants :

- La conception est fondée sur une compréhension explicite des utilisateurs, des tâches et des environnements;
- Les utilisateurs sont impliqués dans la conception et le développement;
- La conception est dirigée et précisée par l'évaluation centrée sur l'utilisateur;
- Le processus est itératif;
- La conception couvre l'expérience de l'utilisateur dans son intégralité;
- L'équipe de conception inclut des compétences et des points de vue pluridisciplinaires.

Deux autres normes s'appliquent pour compléter la conception; ISO16982 [19] se présente sous forme d'un rapport technique comprenant une liste de méthodes s'appliquant aux différentes étapes du cycle de conception centrée sur l'utilisateur, ISO14915 [20] énonce des recommandations générales sur le processus de conception des interfaces multimédias.

La norme ISO9241 découpe en activités le processus de conception centrée utilisateur. Celles-ci sont exécutées de façon itérative, à l'exception de la première (la planification) qui n'est exécutée qu'au début de cycle de développement. La figure 5 illustre la séquence et l'interdépendance des activités de conception centrée sur l'opérateur humain mentionnées par la norme. Nous décrivons brièvement ici les activités.

Planifier le processus

Un projet de conception débute par une phase de planification. Durant celle-ci on définit ce qu'on s'attend à réaliser durant les activités, les personnes responsables des activités et leurs compétences, les procédures de retour d'information, les points de contrôle et les échéanciers, etc.

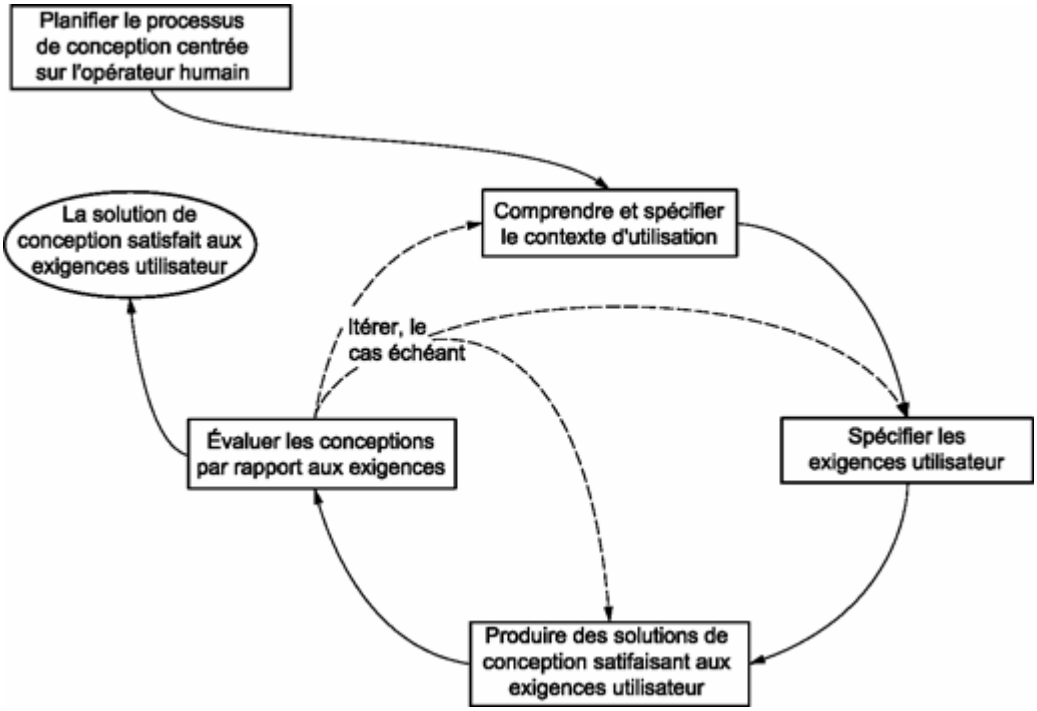


Figure 5 : Interdépendance des activités de conception centrée sur l'opérateur humain selon la norme ISO9241 [17, p.12]

Comprendre et spécifier le contexte d'utilisation

Pour respecter la norme, la description du contexte d'utilisation devrait présenter les caractéristiques suivantes :

- Les utilisateurs et les autres groupes de parties prenantes;
- Les caractéristiques des utilisateurs ou des groupes d'utilisateurs;

- Les objectifs et les tâches des utilisateurs;
- Le ou les environnements d'utilisation du système.

Ces éléments de spécification du contexte permettent de connaître les besoins et contraintes (par exemple; les contraintes d'espace, les exigences de sécurité des utilisateurs, le niveau d'éclairage ou de bruit, un travail effectué en équipe ou de manière autonome, etc.) qui devront être considérés dans la conception du système.

Spécifier les exigences utilisateur

Les exigences utilisateur détaillent les caractéristiques de l'utilisateur qui sont nécessaires à la conception et à l'évaluation des systèmes et qui permettent de répondre aux besoins de l'utilisateur. Les exigences spécifiées peuvent traiter de la performance d'utilisabilité et des critères de satisfaction mesurables. Elles devraient documenter la raison de l'exigence et les compromis acceptables afin d'assurer une compréhension future durant la conception.

Produire des solutions de conception

Les activités précédentes permettent de connaître les caractéristiques et attentes des utilisateurs. Des solutions de conceptions sont élaborées pour répondre à ces caractéristiques. De plus, les solutions produites permettent de prendre connaissance de la répartition des tâches entre le système et les utilisateurs, de concevoir les interfaces utilisateurs et de les communiquer aux autres intervenants de l'équipe de réalisation.

Évaluer la conception

Cette étape est cruciale dans un processus de CCU puisqu'elle permet de faire une évaluation globale des solutions produites afin de juger si les objectifs initiaux sont atteints. Un plan est dessiné pour identifier les éléments recherchés par l'évaluation, les parties du système qui seront évaluées, le artéfacts qui seront utilisés pour l'évaluation (ex., scénarios, prototypes, etc.), le mode d'évaluation (évaluation heuristique par un expert en ergonomie, tests d'utilisabilité avec

utilisateurs, etc.), les informations tirées et leur utilisation dans des étapes subséquentes de la conception. La norme ISO9241-11 [21] énonce des lignes directrices relatives à l'utilisabilité guidant la spécification des critères d'évaluation.

La CCU doit tenir compte des tâches, du contexte et des autres contraintes qui influencent l'utilisateur dans son travail. Pour ceci, les utilisateurs réels et potentiels doivent être considérés. Les utilisateurs réels sont ceux qui utilisent déjà une version de la solution (qu'elle soit informatique ou non) ou qui sont identifiés comme ceux qui utiliseront le produit après sa mise en production. Les utilisateurs potentiels sont des personnes qui ont les mêmes caractéristiques que les utilisateurs réels, mais qui ne font pas usage du système. On peut leur faire appel pour effectuer des tests afin de valider la solution conçue ou développée.

1.2.2 Interface humain-machine

L'objectif de la conception d'interfaces humain-machine (IHM) est de concevoir des dispositifs qui répondent aux besoins énoncés lors des phases d'analyse (analyse des utilisateurs, du contexte, de la tâche). En plus de considérer les besoins utilisateurs, les dispositifs doivent s'adapter à l'architecture de l'application en place.

Les IHM sont un moyen de contrôler et de communiquer avec le système utilisé. Alors, on vise à minimiser l'écart entre le modèle cognitif de l'utilisateur concernant la tâche qu'il tente d'accomplir et les possibilités que lui offre le système pour accomplir cette tâche. Ceci afin d'obtenir un système qui soit considéré ergonomique, qui respecte un certain nombre de critères tels que l'efficacité des interfaces ou la facilité d'utilisation; plus simplement, que le système soit adapté au contexte d'utilisation. La conception d'IHM est basée sur trois principes qui recoupent ceux de la CCU. Ces principes sont les suivants [22] :

- tenir compte de l'utilisateur et la tâche;
- effectuer des évaluations empiriques;
- concevoir de manière itérative.

1.2.3 Utilisabilité

L'utilisabilité, telle que définie dans la norme ISO 9241-11 [21], est le degré selon lequel un produit peut être utilisé par des utilisateurs identifiés pour atteindre des buts définis avec efficacité, efficience et satisfaction, dans un contexte d'utilisation spécifié. L'efficacité est la précision avec laquelle et le degré d'accomplissement selon lequel l'utilisateur atteint des objectifs spécifiés; l'efficience est le rapport entre les ressources requises et la précision et le degré d'accomplissement selon lesquels l'utilisateur atteint des objectifs spécifiés; la satisfaction est l'absence d'inconfort et une attitude positive dans l'utilisation du produit. Enfin, les utilisateurs, leurs objectifs, les tâches accomplies, les équipements utilisés (matériel, logiciel et documents) et l'environnement physique et social dans lequel ils utilisent le produit définissent le contexte d'utilisation.

Différents critères permettent d'évaluer la qualité d'une interface logicielle, tels que la cohérence de l'interface, le guidage, la transparence, etc. Ils n'ont pas tous la même importance selon les projets. Ainsi, il est nécessaire de définir selon quels critères le logiciel sera évalué. L'évaluation du logiciel peut être faite par l'entremise de tests d'utilisabilité. Un test d'utilisabilité, ou test utilisateur, est une technique qui permet de mesurer des problèmes ergonomiques d'un logiciel. Les participants aux tests ont des caractéristiques qui doivent être similaires à celles des utilisateurs visés par le logiciel évalué. Ainsi, pendant qu'un participant réalise un scénario de test avec le logiciel, un ergonome observe son comportement pour constater avec quel niveau de succès le scénario préparé peut être réalisé par le participant. Les comportements du participant servent de base de recommandation de modifications du logiciel pour en améliorer l'utilisabilité.

Nous avons jusqu'à présent décrit les valeurs, les principes et les concepts d'une approche agile de développement logiciel, plus précisément l'approche Scrum. Nous avons présenté les trois thèmes centraux de l'ergonomie logicielle : la conception centrée utilisateur, la conception d'interface humain-machine et l'utilisabilité. Ces thèmes permettent de décrire la base du développement agile et de l'approche de CCU, afin de mieux établir un parallèle entre les activités respectives. Scrum autant que la conception centrée utilisateur sont des approches qui suivent une approche itérative. Le prochain chapitre tente de faire ressortir d'autres similarités

entre les approches ainsi que des défis potentiels à leur intégration. Ces éléments sont utiles afin de pouvoir proposer une méthode qui permet d'intégrer plus aisément des pratiques de l'agilité et de l'ergonomie.

Chapitre 2 RAPPROCHEMENT ENTRE LES APPROCHES ERGONOMIQUE ET AGILE DE DÉVELOPPEMENT LOGICIEL

Ce chapitre fait ressortir les similarités entre les approches ergonomique et agile de développement logiciel afin de pouvoir proposer une méthode simple d'intégration. Le rapprochement est fait autant pour la philosophie sous-jacente à chaque approche que pour le modèle de développement logiciel et des pratiques utilisées. De plus, quelques défis à l'intégration des activités effectuées durant le développement d'un logiciel sont présentés.

2.1 Parallèle entre les activités de Scrum et d'ergonomie

Des thèmes se recoupent entre les approches agile et ergonomique de développement logiciel. Cette similarité entre les deux approches offre un bon point de départ pour l'intégration des pratiques. Les six thèmes suivants semblent les plus pertinents selon les observations de Chamberlain et al. [23] :

1. Implication des utilisateurs : les utilisateurs sont impliqués dans le processus de développement et supportés par l'équipe pour faciliter leurs interventions lors de l'obtention de commentaires (sur les fonctionnalités autant que sur l'utilisabilité), pour comprendre les retours obtenus, pour la rédaction de scénarios et pour s'assurer que les scénarios élaborés contiennent les informations utiles à la conception et au développement.
2. Culture de collaboration : l'équipe formée de concepteurs, de développeurs, d'ergonomes et du client communiquent et travaillent ensemble, de manière fréquente et régulière. Cette communication fréquente permet, entre autres, de faciliter les retours et de faire les ajustements rapides qui limitent les délais aux plans ou concepts élaborés.
3. Prototypage : des prototypes sont réalisés et rapidement ajustés dans un délai qui convient aux intervenants et permet le développement efficace du logiciel.
4. Cycle de vie : un temps suffisant est utilisé au début du projet pour recueillir un minimum d'informations sur les besoins des utilisateurs et le logiciel avant d'amorcer les phases de conception et développement.

5. Gestion de projet : la gestion et le suivi de l'avancement du projet se font de manière à ne pas ajouter d'étapes inutiles et s'ajustent au fur et à mesure de la progression du projet.
6. Conception itérative : le développement est réalisé à partir de données empiriques obtenues lors des retours des cycles précédents. Ceci se manifeste autant par la révision des IHM par les ergonomes, par le remaniement de code (*refactoring*) par les programmeurs ou l'ajustement de la priorité associée à des éléments du carnet de produit par le propriétaire du produit.

Les étapes du modèle de développement logiciel itératif sont similaires à celles du modèle de conception centrée sur l'opérateur humain (figure 5). Nielsen [24] contribue au rapprochement des modèles en précisant que le modèle de développement itératif (figure 6) est utilisé dans la pratique, par les ergonomes. La similarité des deux modèles provient du fait que chacun présente d'abord une étape de planification, suivie d'une étape de spécification puis de conception, de développement d'une solution et enfin de validation de la solution. Ces étapes se déroulent de façon itérative jusqu'à l'obtention d'une solution qui satisfasse les exigences. Cette similitude entre les modèles laisse envisager qu'il y a une intégration possible des activités relevant d'une approche centrée utilisateur et des approches agiles et non pas seulement de la philosophe relative à chacune, comme le font Chamberlain et al.

Quoique les modèles soient similaires, les activités qui découlent de chaque étape sont plus spécifiques à la discipline d'origine. Présentons d'abord les activités propres aux deux disciplines étudiées afin de bien faire ressortir les pratiques similaires, tout comme celles qui sont distinctes. Ceci permettra de connaître des défis rencontrés ou anticipés et de mieux comprendre les enjeux de ne pas intégrer les pratiques d'ergonomie et agiles.

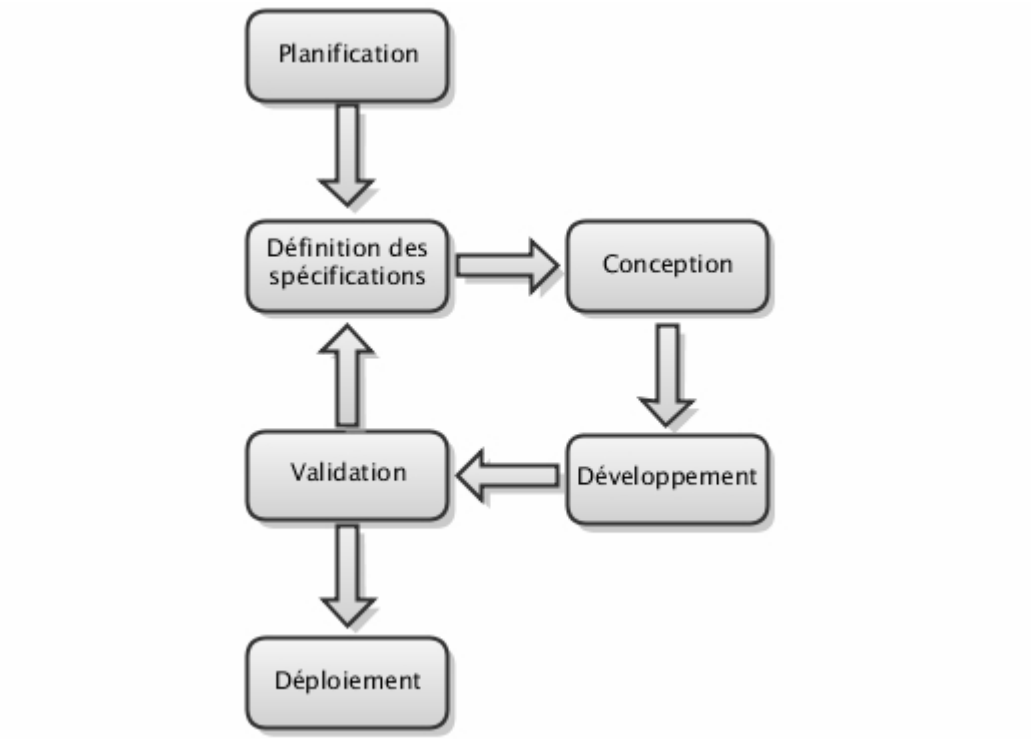


Figure 6 : Modèle du cycle de vie de développement itératif [24].

2.2 Situation actuelle de l'intégration de l'ergonomie aux approches agiles

Il est intéressant et utile d'intégrer les pratiques d'ergonomie à une approche agile de développement logiciel puisque cette dernière ne traite pas spécifiquement des pratiques à utiliser durant les phases de développement. Ces pratiques sont plutôt abordées par des méthodes plus spécifiques telles que Scrum qui décrit des pratiques et cérémonies de gestion de projet comme la mêlée quotidienne et la planification d'itération, eXtreme Programming (*XP*), le développement piloté par les tests (*test driven development*) ou la programmation en paire (*pair programming*). Tout comme ces pratiques, l'application des concepts et des pratiques d'ergonomie logicielle n'est pas décrite dans le contexte d'une approche agile. Pour cette raison, il est utile d'explorer comment ces concepts et pratiques d'ergonomie peuvent s'intégrer à un processus de développement logiciel agile.

Il n'y a pas officiellement une approche d'ergonomie-agile sur laquelle les praticiens des deux domaines s'entendent. Les deux communautés ne sont pourtant pas si éloignées l'une de l'autre. Chacune porte un intérêt à voir que les pratiques s'arriment afin de développer du meilleur logiciel, qui répond autant aux attentes du propriétaire du produit qu'à celles des utilisateurs identifiés. Scott Ambler, auteur et évangéliste des approches agiles, et Jakob Nielsen, guru de l'utilisabilité du web, ont chacun publié des travaux traitant de l'intégration de pratiques d'agilité et d'ergonomie, selon leurs expertises respectives. Ces auteurs offrent des références utiles aux travaux sur le sujet, mais plusieurs éléments restent à démystifier. En se basant entre autres sur les travaux de ces auteurs et afin de mieux comprendre comment intégrer les pratiques d'ergonomie et d'une approche agile à un projet de développement logiciel, voyons d'abord les activités propres à chaque discipline, puis tentons d'en faire un parallèle.

2.2.1 Activités de Scrum

Un projet Scrum débute par une itération de préparation (parfois nommé itération 0), qui dure généralement une ou deux semaines. Cette itération n'est pas officiellement considérée comme une itération du projet de développement logiciel puisqu'elle ne livre pas de valeur d'affaire au propriétaire du produit. Par contre, elle inclut des activités de préparation du projet et est donc fort utile pour lancer le projet dans la direction désirée. C'est lors de cette itération qu'un plan des livraisons (*release plan*) est déterminé, que le carnet de produit initial est estimé et priorisé. L'itération de préparation représente aussi un bon moment pour toute l'équipe de connaître la vision du projet puisqu'on y définit aussi une charte de projet définissant l'objectif du projet, les risques potentiels, les bénéfices prévus et autres informations auxquelles l'équipe peut se référer au cours du projet pour s'assurer qu'elle est toujours en ligne avec la vision initiale. De plus, les critères de succès du projet sont énoncés. Ils permettent de connaître la portée minimale du logiciel pour qu'il soit considéré utile par le propriétaire du produit ou les objectifs spécifiques qui devraient être atteints. Lorsque les critères de succès du projet sont rencontrés, le propriétaire du produit peut alors décider de mettre fin au développement du logiciel. Les projets de développement débutent plus précisément avec l'itération 1. Toutes les itérations subséquentes auront le même déroulement jusqu'à la fin du projet.

À partir du carnet de produit priorisé, l'équipe sélectionne les éléments qu'elle pourra développer au cours de l'itération, en considérant sa capacité de réalisation. La capacité de l'équipe est recalculée après chaque itération pour lui permettre de bien représenter la réalité. Durant l'itération de préparation, puisque l'équipe n'a pas d'historique pour déterminer une capacité moyenne, elle doit en fixer une au meilleur de ses connaissances.

Les éléments du carnet de produit sont rédigés sous forme de récits utilisateurs (*user stories*) et sont généralement présentés sous la forme suivante :

En tant qu'*utilisateur*, afin d'*atteindre un objectif*, je veux *faire une action.*

Dans une approche agile, un récit utilisateur est considéré comme la plus petite quantité d'informations nécessaires pour permettre aux intervenants de définir un chemin dans le système [25]. La brève description de la fonctionnalité désirée contient juste assez de détails pour faire comprendre les besoins et attentes du propriétaire du produit aux autres intervenants du projet. La formulation utilisée permet de rapidement comprendre les besoins d'affaire exprimés par le client et d'estimer l'effort nécessaire pour développer ce récit. L'estimation d'effort contribue à la sélection des éléments du carnet de produit, puisque l'équipe sélectionne les éléments ayant le plus de valeur et qu'elle peut réaliser durant l'itération, selon sa capacité. L'équipe découpe ensuite les éléments sélectionnés en tâches techniques. Un élément du carnet de produit est ainsi découpé pour permettre à l'équipe de modéliser la fonctionnalité associée au récit. Par exemple, le récit :

En tant que *commis à la comptabilité*, afin de *pouvoir effectuer le paiement d'une facture*, je veux *accéder à la facture d'un client.*

Le découpage permettrait alors de faire ressortir la nécessité pour l'utilisateur de;

- s'identifier au système en tant que commis comptable
- faire la recherche d'une facture précise ou d'accéder au profil d'un client pour y retrouver la facture désirée

- payer la facture sélectionnée
- etc.

Le but de ce découpage est de permettre à l'équipe de développement de connaître suffisamment l'objectif de l'utilisateur et les besoins du client dans ce récit pour pouvoir amorcer le développement. Dans un contexte agile, l'équipe ne tente pas de connaître tous les détails de la tâche de l'utilisateur, mais seulement le minimum pour répondre à l'objectif mentionné dans le récit. Au fur et à mesure de l'avancement des itérations, d'autres fonctions seront ajoutées, permettant à cet utilisateur d'effectuer plus d'actions pour rencontrer de nouveaux objectifs. En utilisant des pratiques d'*eXtreme programming*, les développeurs intègrent à chaque itération des courts cycles de programmation et de test. Ils peuvent ainsi être confiants que chaque élément produit respecte les standards de qualité de l'équipe et assurer le client que le logiciel ne contient pas de défauts. Lorsque l'itération est terminée, l'équipe présente le travail complété au propriétaire du produit et obtient ainsi une première rétroaction du travail accompli. Les améliorations et corrections sont alors notées comme des éléments qui seront insérés au carnet de produit.

2.2.2 Activités d'une approche ergonomique

Lorsque les activités d'ergonomie sont intégrées dès le départ à un projet de développement logiciel, il y a d'abord, selon le modèle de Mayhew [26], une phase d'analyse préliminaire. Il est aussi possible d'intégrer les activités à un projet déjà amorcé. Par contre, dans cette situation un sous-ensemble des activités est considéré. Le choix varie selon le contexte du projet et la littérature s'intéresse peu à cet aspect de l'utilisation des pratiques d'ergonomie.

L'analyse préliminaire permet de connaître le profil des utilisateurs visés par le système en conception (par exemple : la fréquence d'utilisation du système, le niveau d'expérience dans le poste occupé, la connaissance des systèmes informatiques, etc.). En effectuant une analyse de tâche de ces utilisateurs, nous pouvons décrire une séquence de travail et des tâches effectuées et

acquérir une meilleure compréhension des objectifs sous-jacents à ces tâches. À partir de ces informations, les critères d'utilisabilité pour évaluer le logiciel à des phases subséquentes pourront être établis. Le travail de conception et de développement qui suit est réalisé de manière itérative. En se basant sur les informations de l'analyse préliminaire, des concepts de haut niveau sont élaborés, comme le flux de navigation, les principales interfaces qui seront développées ainsi que des premières maquettes d'interfaces. Ces dernières représenteront les concepts fonctionnels et l'organisation de l'information permettant une première validation des concepts. Les étapes suivantes de la phase de conception consistent à raffiner les maquettes élaborées. Des validations sont effectuées sur les maquettes, soit par des tests utilisateurs ou des évaluations expertes basées sur des heuristiques choisies pour le projet. D'après les résultats obtenus, les interfaces sont modifiées et détaillées pour être prête à être développée. De plus, ce processus itératif permet de décrire des guides et standards de conception d'IHM qui seront utilisés tout au long du projet. Enfin, la dernière phase permet de recueillir les commentaires d'utilisation du logiciel après installation dans l'environnement du client.

Malgré que des activités des deux disciplines aient le potentiel de se recouper dans leurs objectifs ou d'être effectuées en parallèle, certains défis restent à surmonter. Selon Khatib [27], un grand nombre de praticiens du domaine de l'ergonomie logicielle savent que la CCU dans les projets de développement logiciel qui appliquent une approche agile est peu pratiquée. Alors, une approche tentant de mettre en commun les activités de deux disciplines devra s'assurer de répondre aux défis mentionnés ou anticipés. Avant de mettre en commun les activités pour proposer une méthode d'ergonomie-agile, examinons quelques défis.

2.3 Défis d'intégrer l'ergonomie à l'agilité

Les activités de conception centrée utilisateur, telles que mentionnées dans ISO9241, traitent de méthodes et de techniques de conception ainsi que de leur planification. Cependant, la norme n'aborde pas les aspects de gestion de projet. Pour cette raison, il devient utile d'intégrer les activités de CCU à l'approche Scrum, puisque cette dernière ajoute les concepts de gestion et suivi de projet de développement logiciel. Un objectif de l'ergonomie est d'améliorer l'utilisabilité

d'un logiciel en travaillant sur des activités de conception du logiciel; elle ne traite pas des techniques de réalisation du code. Des pratiques agiles, telles qu'eXtreme Programming, visent plutôt le contraire en tentant d'améliorer les techniques mêmes et ne s'attardent pas aux aspects ergonomiques du développement logiciel. Un défi est d'intégrer ces deux approches pour créer une situation idéale pour les praticiens de chaque approche, mais surtout pour les utilisateurs des logiciels produits [28].

Norman [29] nous fait part d'un défi que les théoriciens de l'utilisabilité ont, en quelque sorte, eux-mêmes posé. Il mentionne que d'un côté ils prônent une approche itérative, adaptative et flexible de conception en favorisant, par exemple, le prototypage rapide. Cependant, les processus mis en place sont plutôt rigides et linéaires et font appel à une conception en amont où les observations des utilisateurs (les tâches, les besoins, le contexte, etc.), la découverte et la prise de connaissance des profils de vrais utilisateurs doivent être effectués avant le début de la conception afin de permettre aux autres étapes de la conception d'être amorcées. Le travail en amont laisse ainsi peu de place à la révision des observations et des profils d'utilisateurs qui ont été dessinés. Cette contradiction, au sein de la communauté, entre une approche itérative-adaptative et une approche de conception en amont doit être résolue afin de faciliter l'adoption des pratiques d'utilisabilité dans un contexte réellement itératif, tel que le développement agile de logiciel.

Un tel compromis entre la conception en amont (*design upfront*) et la conception juste à temps (*just in time*) offre aussi un défi sur une autre facette; effectuer la conception à l'avance nécessite de connaître les besoins et les variables du projet, ce qui s'approche des méthodes traditionnelles. En suivant cette approche, il y a un potentiel de créer des documents ou d'élaborer des concepts inutilement. Alors que dans une approche agile, le carnet de produit peut être modifié en tout temps par le propriétaire du produit. Ainsi, ce qui est considéré comme ayant une grande valeur peut, avec l'apparition de nouveaux éléments dans le carnet, perdre la valeur qui lui est associée. Les éléments qui ont été analysés et qui ont maintenant moins de valeur ou même qui ont été retirés du carnet de produit ont engendré des documents qui n'ont plus d'utilité au projet. Ce genre de perte va à l'encontre d'un principe agile qui spécifie que « la simplicité – l'art de minimiser la

quantité de travail fait inutilement – est essentielle ». Dans la conception juste à temps, qui est très proche d'une approche agile où les documents et les analyses sont réalisés en réaction à la réorganisation du carnet de produit et ce, durant l'itération de développement, il est possible que cette approche nuise à l'équipe de développement qui doit rester en attente pendant que les documents ou prototypes sont élaborés. Le défi est alors de trouver le meilleur compromis entre la conception faite à l'avance qui permet à l'équipe de maintenir son rythme de développement, tout en faisant le minimum de conception pour générer le moins de pertes possible. De plus, le modèle traditionnel tend à vouloir faire la conception du plus d'éléments possibles, tandis qu'une approche agile vise plutôt à faire ce qui est considéré le plus important au moment où la conception a lieu. En s'approchant d'une méthode juste à temps, Sy [30] précise que les itérations futures deviennent ainsi légèrement planifiées, puisque leurs planifications sont basées sur l'information la plus récente à propos de la priorité et de la valeur des éléments du carnet de produit. Par contre, Memmel [31] mentionne qu'un ajustement itératif des IHM, en réaction aux nouvelles demandes du propriétaire du produit, amène une conception qui va à l'encontre des attentes des utilisateurs et diminue la cohérence conceptuelle et navigationnelle des interfaces.

Ambler [32], comme Sy, oriente son discours vers les méthodes adaptatives en ajoutant que les approches ergonomiques de développement logiciel doivent être modifiées pour être plus collaboratives et limiter la quantité de détails inclus dans la conception afin de s'adapter aux cycles de développement agile. Ceci peut être réalisé, entre autres, en ajustant la qualité des prototypes produits ou en effectuant plus tôt des tests d'utilisabilité, lorsque le temps le permet. Il y a d'ailleurs beaucoup à tirer des tests d'utilisabilité puisqu'ils offrent un ensemble de connaissances sur les interactions humain-ordinateur. La conception en amont limite quelque peu les avantages de ces tests. McEvoy [33] ajoute que les tests d'utilisabilité perdent de leur utilité s'il n'y a pas possibilité de tirer profit des connaissances acquises et d'ajuster de manière itérative les interfaces en conséquence.

Les spécialistes de l’ergonomie doivent ajuster leur travail pour au moins fournir une quantité minimale mais utile d'informations aux développeurs. Les équipes de développement doivent aussi ajuster le leur, car les développeurs doivent d'abord être conscientisés du fait qu'ils ne sont

pas eux-mêmes des utilisateurs représentatifs du système qu'ils développent. Ce qu'ils considèrent comme un système simple et utilisable peut ne l'être réellement que pour eux-mêmes et les autres développeurs travaillant sur le même projet. Ce qui n'indique en rien les qualités d'utilisabilité pour de vrais utilisateurs [34]. De plus, selon Ambler [34], les développeurs doivent apprendre à déployer autant d'efforts pour la conception des interactions du système que pour celle de l'architecture technique ou, du moins, reconnaître qu'ils n'ont pas les habiletés pour concevoir des IHM efficaces. En considérant que les développeurs apprennent à faire appel aux spécialistes de l'utilisabilité pour la conception des IHM, le défi provient généralement de l'organisation du travail. Dans certains cas, les ergonomes font partie d'une équipe centrale à laquelle les équipes de développement font appel au besoin. Dans certains autres cas, des ergonomes sont présents dans les équipes de développement. Le choix d'organisation n'est pas simple puisque les deux options offrent des avantages. Par contre, un modèle où les spécialistes en utilisabilité font partie de l'équipe de développement s'approche plus des principes agiles parce que les membres d'une équipe sont co-localisés et partagent le même engagement. Une difficulté liée à ce modèle, selon Nielsen [35], est qu'il n'y ait pas suffisamment d'ergonomes pour répondre aux besoins de toutes les équipes de développement. Une solution proposée pour résoudre ce problème est que les équipes de développement aient un « gardien » des pratiques d'ergonomie. Ce gardien a la responsabilité de s'assurer que l'équipe met en place et respecte les pratiques. En parallèle, les spécialistes en utilisabilité servent de base de connaissances pour les gardiens et sont consultés ou même inclus dans le projet pour répondre aux questions des gardiens lorsque nécessaire.

Un spécialiste de l'ergonomie qui est à l'extérieur de l'équipe de développement a le désavantage de ne pas avoir le même niveau d'engagement que les autres membres envers la livraison pour l'itération. De plus, il y a un fossé qui se creuse entre les « communautés » lorsqu'il y a d'un côté l'équipe de développement et de l'autre le spécialiste d'ergonomie. Plusieurs autres défis doivent être surmontés afin que les deux communautés travaillent effectivement ensemble pour créer du meilleur logiciel, et ce, de façon commune. Au delà des pratiques, reste le défi philosophique parce que le développement logiciel n'est pas abordé du même angle. Le tableau 1 présente des différences de philosophie entre les méthodes agile et de l'ergonomie.

Tableau 1 : Comparaison entre les philosophies des méthodes agile et de l'ergonomie [34].

Philosophie agile	**Philosophie ergonomique**
Comment améliorer le système au cours de cette itération?	Quel est le système idéal (pour l'utilisateur)?
Les détails de spécifications peuvent être identifiés juste à temps, durant la phase de développement. La conception détaillée en amont est risquée.	Les comportements du système doivent être identifiés en amont, avant le début de la phase de développement.
Centrer la conception sur les besoins énoncés par le propriétaire du produit.	Centrer la conception sur les besoins et les attentes de l'utilisateur.
Miser sur des pratiques de conception technique de système.	Mettre l'accent sur la conception d'un système qui peut être utilisé de manière efficiente.

Il est aussi nécessaire de sensibiliser le propriétaire du produit à l'importance des pratiques de l'ergonomie afin d'obtenir un logiciel utilisable. Cette sensibilisation doit se faire selon deux volets. L'un est d'accorder de l'importance aux éléments de carnet traitant spécifiquement de critères d'utilisabilité [36]; par exemple, en créant des récits utilisateurs qui spécifient clairement un objectif d'utilisabilité, tel que « En tant que *commis à la comptabilité*, afin de *réduire la possibilité d'erreur en effectuant le paiement d'une facture*, je veux *bien voir le nom et le numéro de la facture que je tente de payer lorsque j'accède à la facture d'un client* ». Dans cet exemple, le propriétaire du produit peut évaluer la valeur de ce récit en considérant l'importance de la réduction d'erreur pour le commis à la comptabilité.

L'autre volet de sensibilisation du propriétaire du produit est d'ajuster le modèle proposé pour respecter l'investissement du client. Une valeur de l'agilité est l'adaptation au changement. Alors, il doit être possible d'adapter une approche d'ergonomie-agile aux attentes du client. Bien qu'il soit idéal d'effectuer toutes les activités proposées, pour certains clients il est nécessaire de sélectionner un sous-ensemble, ceci à cause de limitation des délais ou du budget associé au projet. Quelles sont celles qui apportent le plus de valeur dans le contexte et selon les attentes du client? Quel sous-ensemble de pratiques est-il possible d'utiliser pour s'assurer que le logiciel

réalisé soit utilisable par les utilisateurs visés? Le modèle de Sy, entre autres, ne s'attarde pas à cette problématique et assume que toutes les activités proposées seront réalisées. Or, il a été constaté dans certains projets qu'il n'est pas possible d'exécuter toutes les activités proposées, et ce, d'après des contraintes imposées par le client. Le guide qui sera proposé au chapitre 4 tiendra compte de ces questions, afin de s'assurer qu'il offre assez de flexibilité aux équipes de développement désirant utiliser une approche d'ergonomie-agile.

Ces éléments ne représentent que certains des principaux défis qui sont à surmonter. Il en existe d'autres dont l'impact varie selon les projets où une intégration des pratiques d'ergonomie à une approche agile est tentée. Avant de proposer une méthode d'intégration des pratiques d'ergonomie et agiles, examinons dans le tableau 2 une liste des principaux problèmes rencontrés par des ergonomes en appliquant des pratiques agiles, les symptômes associés et des pistes de solutions. Ceci nous permet déjà d'entrevoir des solutions possibles aux défis mentionnés et d'utiliser des solutions proposées.

Tableau 2 : Résumé des principaux problèmes rencontrés par des ergonomes qui appliquent des pratiques agiles [37].

Problèmes	**Symptômes**	**Solutions possibles**
Temps insuffisant pour la conception	• Les développeurs attendent après les concepts • La qualité de conception diminue • Les concepts ne sont pas vérifiés avec les utilisateurs	• Activités distinctes et parallèles d'ergonomie et de développement • Activités d'ergonomie ayant une portée courte et incrémentale • Test d'utilisabilité avec une méthode à rabais • Conception contextuelle rapide • Découper les éléments à concevoir • Combinaison de différentes activités d'ergonomie en une session • Amener un utilisateur dans le projet
Itérations sont trop courtes	• Les concepts ne peuvent être terminés à temps • Manque de temps pour les tests d'utilisabilité • Manque de temps pour la communication avec le client	
Manque de commentaires/ rétroactions des utilisateurs	• Les rétroactions arrivent trop tard • Aucune donnée pour agir – seulement des opinions • Le produit n'est pas validé	

« Client » agile absent	• Le client final ne s'implique pas (dans le processus) • Impossible d'obtenir l'adhésion du reste de l'équipe • Des décisions non informées sont prises	• L'ergonome peut agir comme client agile • Chaque ergonome travaille dans une équipe agile • Choisir judicieusement les membres de l'équipe agile • Les concepts validés sont transférés aux développeurs pour être réalisés • Les ergonomes participent à la planification d'itération pour apporter des commentaires relatifs aux utilisateurs
Ergonome pas à temps plein dans l'équipe	• Les ergonomes sont occupés dans des rencontres et non à faire du travail pour l'itération • Baisse de moral causé par la diminution de la qualité ergonomique	
Aucune planification d'itération	• Grande quantité de défauts dans le carnet de produit • Les commentaires sur les priorités sont ignorés • Pas de contrôle sur les délais ou l'estimation	
Retours/ rétroactions des utilisateurs ignorés	• La portée des fonctionnalités est fixée et non modifiable • Pas de temps alloué pour intégrer les changements • Aucune réorganisation de priorité permise	
Manque d'une vue globale	• Aucune vision commune • Trop grande concentration sur les détails • Difficulté à établir des priorités/ prendre des décision de conception	• Convaincre l'équipe de compléter une itération de préparation • Alléger le processus de collecte des exigences • Raccourcir la portée des décisions à 1 ou 2 itérations d'avance
Mauvaise communication	• Concepts mal compris • L'équipe agile n'adhère par aux concepts • L'information importante est perdue	• Inclure les développeurs dans le processus de conception • Inclure les critères d'utilisabilité dans les conditions d'acceptation • Vérifier le progrès quotidiennement • Les documents servent à l'équipe de conception

L'équipe n'est pas colocalisée	• Pas d'esprit d'équipe • Barrière de langage s'installe • Manque de communication entre les membres	• Utiliser des outils de télétravail • Colocalisé les membres de l'équipe pour la planification
Problèmes de dépendances	• Besoin d'éléments de travail venant de l'extérieur de l'équipe agile • Difficulté à terminer les tâches à temps pour achever l'itération	• Faire appel à un facilitateur persuasif pour faire avancer rapidement les activités

Nous avons décrit les principales activités de l'approche Scrum et de l'ergonomie du logiciel. Ces connaissances ont permis d'établir les parallèles et les similarités qui existent entre les approches, afin de mieux comprendre le potentiel d'intégration des activités des deux domaines. Bien qu'il y ait plusieurs éléments de rapprochement, il y a des défis à relever pour réussir une intégration. Dans le prochain chapitre nous examinons les approches d'intégration proposées par deux auteurs et présentons par la suite les concepts menant à la conception du guide d'intégration. Ces éléments contribuent au rapprochement des approches des auteurs vers les principes derrière Scrum et tentent de surmonter les défis mentionnés.

Chapitre 3 INTÉGRER L'ERGONOMIE À L'APPROCHE AGILE

Il y a plusieurs méthodes possibles pour intégrer les approches d'ergonomie et agile de développement logiciel. Il ressort généralement deux propositions de fonctionnement pour l'intégration des activités : effectuer les activités d'ergonomie en pré-phase au développement ou effectuer les activités d'ergonomie à l'intérieur des itérations Scrum [28]. La première proposition signifie que l'équipe d'ergonomes effectue du travail un ou plusieurs cycles avant que l'équipe de développement implémente les concepts ainsi réalisés, ceci afin que les développeurs puissent amorcer leur travail sur des concepts complets et validés. À l'opposé, la deuxième proposition indique que les ergonomes et les développeurs travaillent ensemble, durant chaque cycle du projet, à réaliser des concepts et à développer les fonctionnalités reliées à ces concepts.

Le principe sous-jacent à la première proposition va en quelque sorte à l'encontre de principes de l'agilité. Selon une approche agile, il n'est pas souhaitable de couvrir tous les aspects et faire ressortir toutes les spécifications d'un produit au début. Agile prend surtout une approche acceptant que les spécifications changent et évoluent au cours du projet. Tandis que dans la deuxième proposition, les considérations ergonomiques deviennent liées de trop près aux détails techniques d'implémentation et de construction du logiciel. Les pratiques d'ergonomie permettent de cadrer la conception des interactions de l'utilisateur et du logiciel, et ce, à partir du point de vue de l'utilisateur et non en se basant sur des décisions d'implémentation [28].

Pour contrer les faiblesses mentionnées, adopter une approche combinant le fonctionnement suggéré par ces deux propositions semble la plus prometteuse. Des auteurs comme Nielsen [24] et Sy [30] proposent des méthodes qui vont dans ce sens. Nous allons prendre connaissance des propositions de ces deux auteurs afin de mieux découvrir les points communs, de connaître les avantages des pratiques présentées ou les problèmes perçus durant des projets de développement logiciel.

3.1 Modèles de travaux d'ergonomie durant les cycles de développement

À travers des entrevues et des expériences, Nielsen (2008) et Sy (2007) ont pu chacun élaborer un modèle d'activités d'ergonomie dans un contexte agile (ou Scrum). La figure 7 présente le modèle élaboré par Nielsen d'après les résultats de sondages auprès de professionnels du développement logiciel.

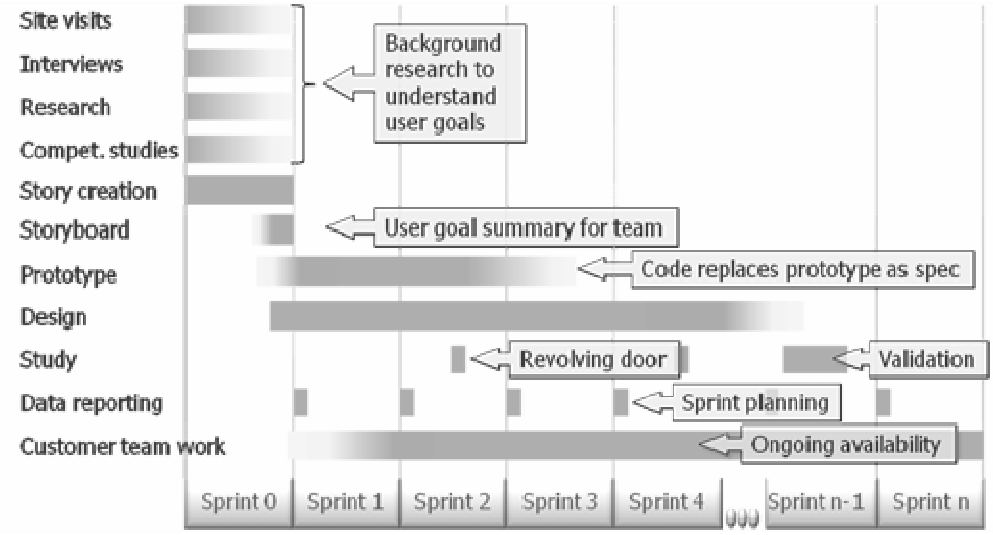


Figure 7 : Répartition du travail d'ergonomie durant les itérations Scrum [23, p.30].

Dans le modèle élaboré par Nielsen, l'itération 0 (itération de préparation) sert à recueillir une partie des informations nécessaires au démarrage du développement du projet. Malgré que dans une approche agile on préfère laisser émerger les besoins et les spécifications au cours du projet et laisser la possibilité de faire évoluer les informations déjà acquises, il est fort utile de recueillir une quantité d'informations sur les utilisateurs, leurs tâches et leur contexte de travail avant le début du développement. Ces informations sont utiles pour la création des scénarios d'utilisation afin qu'ils soient orientés selon la réalité des utilisateurs. Elles peuvent être obtenues en combinant les données recueillies durant des observations des utilisateurs au travail et des entrevues avec ces derniers, la recherche préliminaire (*background research*) ou une analyse compétitive. Dans une approche agile, l'itération de préparation permet de faire ressortir la vision

du projet et les grands besoins tels que perçus par le propriétaire du produit. L'implication d'un ergonome à l'itération de préparation contribue à créer et utiliser des scénarios pour construire des flux de tâches et des scénarimage (*storyboard*). La création des scénarimages a lieu durant l'itération de préparation pour donner un aperçu du logiciel entier, afin de créer une certaine cohérence de navigation permettant l'accomplissement des tâches par l'utilisateur. À chaque itération subséquente, l'équipe d'ergonomes améliore les scénarimages pour y incorporer les éléments qui seront développés durant l'itération suivante par l'équipe de développement. Ces scénarimages deviennent ainsi une forme de documentation vivante du projet. Ils sont considérés comme vivants puisqu'ils évoluent et suivent l'avancement du projet. Pour suivre la philosophie agile, les ergonomes doivent travailler de pair avec l'équipe de développement. Alors, leur travail est divisé entre la préparation des itérations à venir et le support à l'équipe de développement pour l'itération en cours. Il doit donc y avoir une gestion du travail qui est fait en parallèle. Ainsi, les itérations de développement et celles comportant le travail d'ergonomie se déroulent selon le même échéancier et sont suivies de la même manière. La proposition de Nielsen est intéressante mais demeure quand même de haut niveau quant aux activités à effectuer durant les itérations. Ainsi, il devient difficile pour les équipes de définir la portée du travail d'ergonomie à accomplir durant une itération. Le modèle proposé par Sy à la figure 8 reprend des concepts proposés par Nielsen mais tente d'être plus concret.

Durant l'itération de préparation, nommée Sprint 0 par Nielsen (figure 7) et Sy (figure 8), le travail d'ergonomie consiste à recueillir les données pour définir l'étendue et l'objectif du logiciel, effectuer des visites auprès des utilisateurs pour valider les données déjà connues, analyser les données existantes et décrire des profils d'utilisateurs et de flux de processus (en utilisant des personas et des scénarios) pour modéliser, de haut niveau, les systèmes et les interactions possibles. Lors de la première itération, l'équipe de développement débute par la programmation des éléments qui ne nécessitent pas d'interface utilisateur (par exemple : traitement en lot d'un ensemble de données, conversion automatique de fichiers, mise en place de fonction de synchronisation de base de données, etc.). Durant la première itération, un travail de conception est fait pour élaborer les maquettes et les récits et effectuer des tests d'utilisabilité sur les concepts qui ont été réalisés durant l'itération de préparation. Alors, puisque les IHM conçues par l'équipe d'ergonomes n'ont pas été validées, l'équipe de développement ne peut pas réaliser,

à ce moment, des éléments qui nécessitent des interfaces utilisateurs. Aussi, dans le but de préparer la deuxième itération pour l'équipe de développement, un travail similaire d'analyse des utilisateurs qui a été fait durant l'itération de préparation est fait pour préparer le travail des ergonomes pour l'itération suivante. Ce cycle se répétera ainsi pour les itérations suivantes.

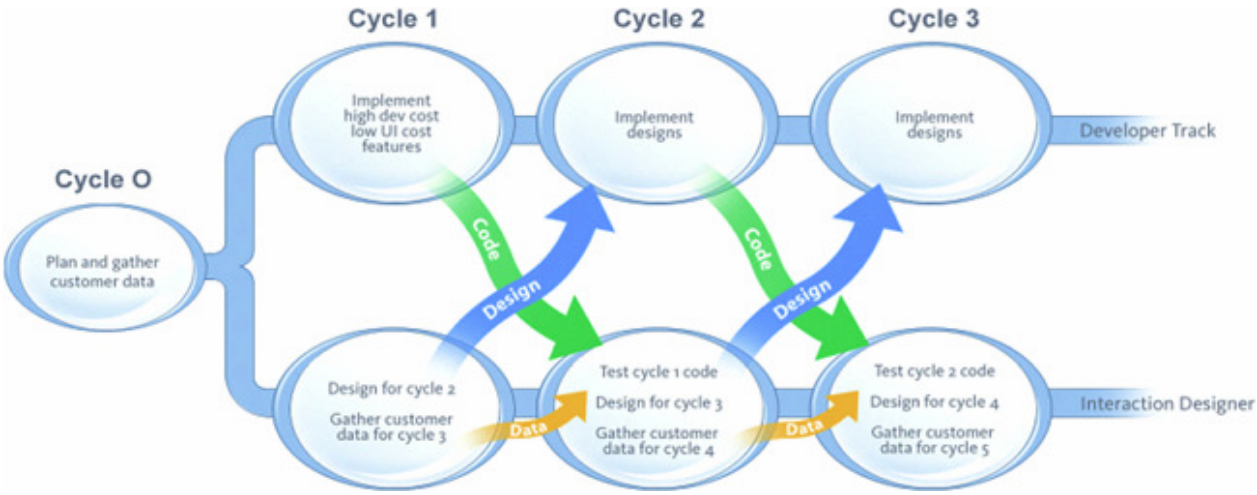


Figure 8 : Représentation du travail parallèle d'ergonomie et de développement agile [29, p118].

Les modèles proposés par Nielsen et Sy sont efficaces et permettent d'ajuster les concepts rapidement en obtenant des retours de l'équipe de développement sur la capacité de produire les concepts, et aussi de valider auprès d'utilisateurs les concepts qui ont été réalisés. Dans une approche plus traditionnelle, telle une approche en cascade, ces rétroactions ne viendraient généralement que très tard dans le processus de développement et plusieurs mois, voire années, après l'élaboration des concepts, puisque le processus de développement est décomposé en longues phases séquentielles, qui laissent très peu de place aux ajustements suite aux rétroactions obtenues.

Les modèles présentés aux figures 7 et 8 sont cohérents entre eux mais laissent un certain nombre de questions sans réponse ce qui peut nuire à l'adoption des pratiques d'ergonomie au sein d'équipes agiles. Voici ces questions :

- Durant les itérations « d'ergonomie », comment l'équipe fait-elle le découpage en tâches, et comment calcule-t-elle sa capacité? Le découpage en tâches est important pour rendre visibles à tous les intervenants du projet les détails du travail qui est prévu pour la prochaine itération, tandis que la capacité est nécessaire pour connaître la quantité de travail que l'équipe peut réaliser.
- Comment l'équipe d'ergonomes rend-elle visibles les tâches qu'elle prévoit accomplir durant l'itération? Les équipes de développement agile utilisent traditionnellement un carnet d'itération pour connaître et effectuer le suivi quotidien des tâches et leur avancement.
- Est-ce que les équipes d'ergonomes doivent aussi avoir en parallèle un carnet d'itération qui est déconnecté de celui de l'équipe de développement? Cette approche offre une bonne visibilité aux tâches d'ergonomie, par contre, en les affichant dans un carnet spécifique à l'équipe d'ergonomie, elle ramène le problème d'avoir deux équipes qui travaillent en parallèle, sans avoir le même objectif de réalisation pour l'itération en cours.

Pour l'équipe de développement, les méthodes proposées par Sy et Nielsen laissent aussi leur part de questions.

- Comment l'équipe de développement aborde-t-elle les récits et les maquettes élaborés durant l'itération précédente? Les récits et maquettes couvrent parfois plus de détails que ceux mentionnés dans un élément du carnet de produit. Il devient alors difficile de faire un suivi adéquat de l'avancement des tâches de développement.
- La validation auprès des utilisateurs identifiés par les ergonomes devrait-elle se faire à chaque itération de développement ou seulement selon des intervalles plus grands? Des rétroactions fréquentes permettent d'assurer le développement d'un système conforme aux attentes des utilisateurs. Par contre, les tests utilisateurs nécessitent d'avoir un groupe d'utilisateurs assez nombreux pour effectuer les tests, afin d'éviter de réutiliser les mêmes personnes trop fréquemment et ainsi de perdre leur objectivité face au système développé. De plus, leur fréquence ne doit pas nuire à l'avancement des travaux.

- Comment l'équipe de développement doit-elle traiter les résultats obtenus des validations faites par les ergonomes? Les résultats peuvent être traités comme des nouveaux éléments insérés dans le carnet de produit et qui doivent être priorisés par le propriétaire du produit ou considérés comme des défauts à corriger au même titre que des problèmes techniques rencontrés lors de tests fonctionnels. Alors, les corrections d'utilisabilité sont mises en priorité dans l'itération suivante puisqu'elles ne répondent pas correctement au comportement désiré du logiciel.

3.2 Méthode d'intégration

La méthode d'intégration proposée tente de répondre aux questions précédentes tout en respectant la philosophie sous-jacente aux modèles de Nielsen et Sy. Cependant, la méthode fait ressortir un élément important, le travail d'ergonomie et le travail de réalisation technique du logiciel sont faits par une seule équipe. La distinction entre les développeurs et les ergonomes est réduite, en tentant même de la faire disparaître. Ceci représente certes un défi mais il n'est pas insurmontable. D'abord, pour mieux présenter les améliorations des méthodes proposées, découpons les activités selon l'itération ou la phase dans laquelle elles se produisent et voyons ensuite comment elles peuvent être abordées durant une itération.

Les ergonomes, tout comme les développeurs, font partie de l'équipe Scrum. Ceci est cohérent avec ce qu'Aigner [28] avance ainsi qu'avec les expériences acquises par l'auteur dans le cadre de projets de développement logiciel. Le fait d'inclure les ergonomes dans l'équipe de développement leur permet d'avoir le même niveau d'engagement de livraison pour l'itération. Les ergonomes sont aussi en quelque sorte membres de « l'équipe » du propriétaire du produit et contribuent à la décision de la valeur d'affaire des éléments du carnet de produit [38] et travaillent à le sensibiliser afin qu'il associe de la valeur au travail et aux tâches d'ergonomie telles que la description des caractéristiques des utilisateurs au début du projet, la conception de récits et de maquettes durant chaque itération, la validation auprès d'utilisateurs à certains moments critiques du développement, etc. Ambler [34] abonde dans ce sens et ajoute que l'ergonome travaille avec l'équipe de développement sur les tâches de l'itération en cours.

Itération de préparation et Analyse préliminaire

L'itération de préparation est critique puisqu'elle permet de découvrir et de construire la base du projet. Les activités d'ergonomie (analyse préliminaire) devraient débuter durant l'itération de préparation de Scrum. Le travail d'analyse de tâches et d'entrevues avec les utilisateurs visés permettra de mieux cibler les récits utilisateurs qui seront élaborés durant cette itération. Alors, pour la première phase du projet, les activités réalisées devraient permettre de faire ressortir certaines informations, utiles pour la suite du projet :

- Le comportement des utilisateurs visés lorsqu'ils effectuent leur travail. Ceci afin de reproduire le comportement attendu dans le logiciel;
- Les caractéristiques des utilisateurs obtenues à partir des données d'observations des utilisateurs. L'itération de préparation est un bon moment pour faire une première élaboration des personas, par exemple, tel que mentionné par [38] et observé lors de la conception d'un logiciel permettant l'administration d'un système de résonance magnétique;
- Élaborer des scénarios d'utilisation généraux qui permettent de faire une représentation sommaire du système, à partir des informations connues à ce stade de réalisation du logiciel.

Les activités précédentes présentent un niveau de détail assez global et risquent peu de changer durant le processus de conception et développement du projet [28]. Alors, malgré que dans les approches agiles on préfère effectuer les analyses le plus tard possible dans le processus de développement, il est stratégique de faire ressortir certaines informations le plus tôt possible, ceci afin d'avoir, dès le début du projet, une entente sur le profil, les tâches, les objectifs et les attentes des utilisateurs envisagés pour le logiciel en conception. Ce sont ces informations qui aideront le propriétaire du produit à composer et prioriser son carnet de produit.

Connaissant globalement la portée et les objectifs du logiciel à développer, l'équipe de développement est alors en mesure d'effectuer des choix technologiques qui répondront le mieux

au système anticipé et de préparer son environnement de travail afin d'être prête à commencer la programmation à l'itération 1.

L'itération de préparation permet aux équipes agiles de déterminer les critères d'évaluation d'un élément du carnet d'itération pour pouvoir le considérer comme terminé. Puisque des ergonomes sont maintenant présents dans l'équipe, il est possible aussi d'ajouter des considérations ergonomiques aux conditions menant à une tâche terminée (*definition of done*).

Un objectif de l'itération de préparation est de s'assurer que le travail de l'itération 1 peut être amorcé, et ainsi, commencer la réalisation du logiciel. Alors, en plus de ce qui est déjà mentionné, d'autres activités permettent de concrétiser l'information recueillie :

- À partir des scénarios d'utilisation généraux, construire un modèle de navigation du système afin de permettre à tous les intervenants de connaître, de manière générale, la répartition à travers le système des fonctionnalités mentionnées dans les scénarios. Ceci offre aussi l'avantage de pouvoir constater si un scénario est trop général et englobe une grande partie du système ou s'il demeure localisé à une partie identifiable;
- Rédiger les récits utilisateurs qui peuvent être extraits des scénarios d'utilisation. Ceci constitue la première version des éléments du carnet de produit. En faisant appel à des récits utilisateurs pour constituer le carnet de produit, le propriétaire du produit se donne une certaine assurance que les éléments du carnet sont orientés utilisateur;
- Prioriser le carnet de produit;
- Estimer le travail à faire pour réaliser les éléments du carnet, autant le travail technique que le travail d'ergonomie;
- Élaborer des maquettes d'interfaces et d'interactions pour les éléments du carnet qui seront développés durant l'itération 1. Il est possible qu'une maquette couvre plusieurs éléments du carnet. Il est alors important de bien identifier quels éléments de la maquette correspondent à quels éléments.

Ces étapes font partie de l'itération de préparation. Chacune des étapes devrait être réalisée et inclure tous les membres de l'équipe, permettant à chacun de bien comprendre la portée des premiers éléments du carnet de produit et de s'approprier l'approche utilisée.

Itération 1/Conception et développement

Pour bien fonctionner avec une approche Scrum, le travail à faire doit être planifié pour l'itération. Le travail à planifier et réaliser durant l'itération concerne l'équipe complète, alors il est nécessaire de considérer :

- La réalisation technique des récits utilisateurs. Des éléments du carnet de produit sont développés et deviennent ainsi des incréments fonctionnels de logiciel;
- L'élaboration et la validation de maquettes d'interfaces. Ces activités relevant d'une approche de CCU permettent de prendre des récits utilisateurs et les analyser. L'analyse faite est limitée aux détails nécessaires pour la réalisation de ce récit. Les récits ainsi analysés pourront être réalisés à l'itération suivante. Ce fonctionnement ressemble, en quelque sorte, à une approche en cascade. Cependant, la portée de ce qui est analysé est limitée à un court récit plutôt qu'au logiciel entier. L'utilisation de maquettes augmente la compréhension du récit utilisateur par rapport à une description purement textuelle. Comme le mentionne [25], il est possible qu'en utilisant une représentation textuelle, les développeurs et les autres parties prenantes pourraient chacun avoir une compréhension différente d'un récit utilisateur;
- La création de nouveaux éléments du carnet de produit par le propriétaire du produit. Le propriétaire du produit continue d'alimenter le carnet de produit avec de nouveaux récits utilisateurs, qu'il priorise au fur et à mesure, afin que l'équipe puisse être prête, le temps venu, à effectuer l'itération suivante.

Pour souligner que les ergonomes et les développeurs travaillent au sein de la même équipe et s'engagent à la réalisation du même projet, tous les travaux sont notés au même carnet de produit. D'un côté, ceci permet au propriétaire du produit de bien prendre conscience de l'importance de

tous les travaux. De plus, durant les cérémonies de planification et présentation du travail accompli, tous les membres de l'équipe sont présents pour constater le travail réalisé.

Itération 2 à n-1/Conception et développement

À partir de la deuxième itération, les activités relevant d'une approche de CCU s'étendent; puisque l'équipe a réalisé un incrément fonctionnel de logiciel à l'itération précédente, des activités permettant d'obtenir des rétroactions de la part des utilisateurs sont aussi à prévoir dans le travail de l'itération. Suite à ces rétroactions, des commentaires et améliorations sont proposés. Ils sont insérés comme des nouveaux récits utilisateurs dans le carnet de produit.

Nous avons pris connaissance de deux approches qui existent pour intégrer des activités d'ergonomie à celles de Scrum et avons aussi présenté le principe sous-jacent au guide que nous avons réalisé. Le prochain chapitre présente le guide avec détails pour permettre de rattacher tous les concepts qui ont été mentionnés dans ce travail jusqu'à présent.

Chapitre 4 GUIDE D'INTÉGRATION DE L'ERGONOMIE À L'APPROCHE AGILE

Pour pouvoir concevoir ce guide d'intégration, il a été nécessaire de bien retracer les éléments de base qui définissent une approche de CCU ainsi qu'une approche agile de développement logiciel. Ce sont les différents travaux d'auteurs, théoriciens et praticiens en plus de notre propre expérience de travail en matière de CCU et de développement agile dans de nombreux projets qui ont permis de faire avancer la conception d'un tel guide. Les travaux de Mayhew [26], Sy [30] et de l'auteur sont d'autres sources permettant de répondre à des interrogations posées en introduction de ce travail. Nous décrivons dans ce chapitre la liste les activités présentes dans le guide d'intégration que nous avons développé; le guide complet est présenté à l'annexe 1. Aussi, nous présentons la méthodologie d'évaluation du guide, les résultats et les améliorations portées au guide suite à une première phase d'évaluation. D'autres phases d'évaluation sont à prévoir pour continuer à améliorer le contenu du guide.

4.1 Présentation du guide

Le guide présenté dans ce travail est le résultat de deux itérations. Une première a servi à mettre en place les activités du guide d'après les travaux recensés d'autres auteurs et de l'expérience de l'auteur en développement logiciel selon une approche Scrum. La seconde itération a puisé dans les commentaires et les suggestions d'améliorations provenant d'une première phase d'évaluation du guide. Cette phase d'évaluation est décrite à la prochaine section.

Dans le guide, il est proposé d'ajouter aux cérémonies de Scrum certaines activités plus spécifiques à l'ergonomie. Le guide propose 18 activités distinctes, provenant autant d'une approche Scrum que de CCU. Le choix des activités de CCU proposées correspond, entre autres, à des activités décrites dans des travaux de Hussain [39], Schwartz [40], Sy [30] et Nielsen [24]. Les activités de CCU suivantes ont été sélectionnées parce qu'elles étaient mentionnées par plusieurs de ces auteurs et semblaient être les plus pertinentes et applicables dans un contexte Scrum de développement logiciel :

- Prototypage à basse résolution d'interfaces;
- Analyse de tâches des utilisateurs;
- Tests d'utilisabilité;
- Conception de scénarios;
- Description des utilisateurs visés;
- Élaboration de personas.

Les activités proposées pour chaque itération sont présentées dans l'ordre d'exécution recommandé. Cet ordre semble logique à l'intérieur de chaque itération et favorable au bon déroulement du projet. Le choix de la séquence d'exécution des activités de Scrum suit les pratiques habituelles de développement logiciel. Cependant pour les activités de CCU, il est nécessaire de questionner l'ordre habituellement utilisé. Nous devons mettre en place une séquence qui apporte de la valeur au processus de développement, sans ajouter de lourdeurs et s'assurer de respecter les principes d'une approche Scrum. La séquence des activités de CCU est insérée dans celle de Scrum pour ajouter, au moment jugé opportun, les détails nécessaire à l'avancement du projet. Elle suit un déroulement qui permet de préciser les détails acquis durant une phase précédente de la conception, en apportant un minimum d'informations utiles à l'activité en cours, sans toutefois tenter de concevoir tout le système ou pour tous les cas d'utilisation possibles. La description des utilisateurs, par exemple, décrit les utilisateurs d'après les caractéristiques utiles au développement en cours et ne tente pas de faire ressortir toutes les caractéristiques des utilisateurs. Les informations recueillies durant l'analyse de tâche se limitent aussi à ce qui est connu pour une tâche spécifique et non le travail entier de l'utilisateur. Ceci afin de permettre que les activités de CCU soient réalisables dans le cadre d'une itération Scrum et qu'elles ne produisent pas un surplus inutile d'information. De plus, il est possible à chaque itération d'ajouter de nouvelles informations à celle connues, à mesure que la connaissance du système ou des utilisateurs augmente.

Il n'y a pas de distinction faite entre les activités de CCU et celles relevant de Scrum. Ceci afin de favoriser la pleine intégration des pratiques de travail relevant des deux domaines. En effet, puisqu'une équipe pluridisciplinaire de Scrum a en principe les compétences nécessaires pour réaliser le produit désiré par le propriétaire du produit, il n'est pas nécessaire d'ajouter la distinction du domaine d'origine. Ainsi, l'équipe peut s'approprier toutes les activités peu importe de quel domaine elle relève.

4.1.1 Liste des activités

Le guide propose la liste d'activités suivantes et les représente dans le schéma présenté à la figure 9. Elles seront par la suite présentées selon l'itération où elle devrait apparaître.

(01) Prendre connaissance du projet, de la vision du propriétaire du produit et des objectifs du logiciel

Le PP transmet à l'équipe de développement sa vision et l'objectif qu'il tente de réaliser avec le logiciel qui sera développé. Ces éléments permettent aux membres de l'équipe de prendre des décisions éclairées lors de la conception et du développement du logiciel pour produire un logiciel qui respecte les attentes premières du PP. De plus, en connaissant l'objectif du système, l'équipe peut avoir de meilleures conversations avec le PP et même le questionner sur la pertinence de certains récits utilisateurs.

(02) Analyser le contexte actuel

Le contexte actuel d'utilisation représente les connaissances sur le travail tel qu'il est effectué par les utilisateurs. Au moment de l'analyse du contexte, le travail peut être effectué d'une manière entièrement différente de celle envisagée dans le logiciel qui sera développé. C'est à partir de cette analyse qu'il est possible de faire ressortir les caractéristiques du travail en cours, les systèmes utilisés, les contraintes existantes qui influent sur le déroulement du travail.

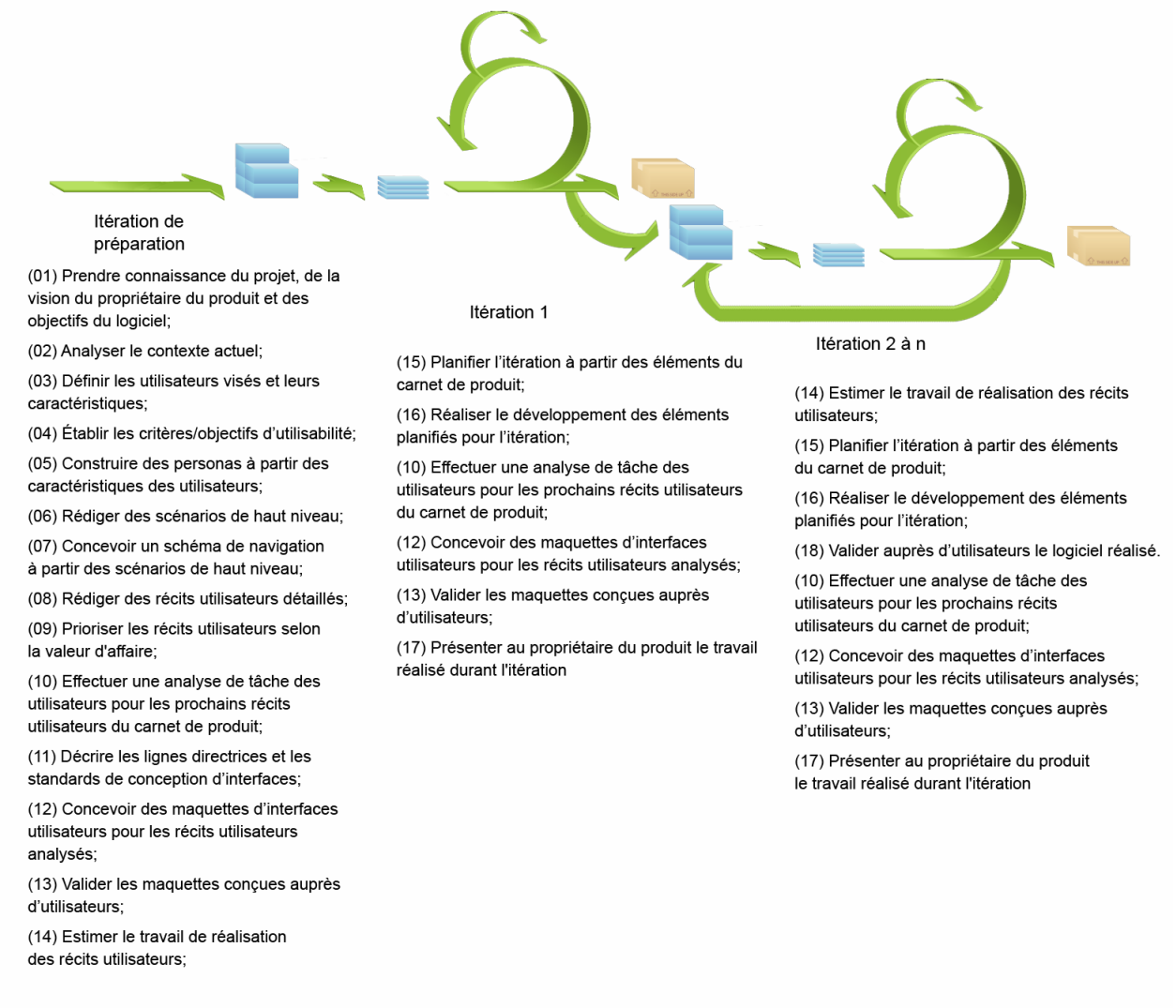


Figure 9 : Schéma des activités d'intégration de la CCU à une approche Scrum

(03) Définir les utilisateurs visés et leurs caractéristiques

L'équipe travaille avec le PP pour identifier les utilisateurs visés par le logiciel. Une partie de cette information provient de la vision et des objectifs du logiciel, puisqu'en élaborant sa vision du logiciel, le PP est amené à définir le public visé. Il s'agit de décrire les utilisateurs visés de manière assez détaillée mais sans pour autant être exhaustif. Ceci facilitera la prise de décision dans la majorité des activités qui suivent.

(04) Établir les critères/objectifs d'utilisabilité

Les critères d'utilisabilité permettent d'établir comment sera évaluée l'utilisation du logiciel. Ils sont déterminés à partir des caractéristiques des utilisateurs (motivation, expérience à exécuter le travail, fréquence d'exécution, caractéristiques physiques, etc.) afin que les utilisateurs puissent tirer plein potentiel du logiciel. Les critères d'utilisabilité (temps d'exécution d'une tâche, fréquence des erreurs, degré d'autonomie de l'utilisateur, niveau de satisfaction, etc.) guideront la conception du logiciel tout au long du développement.

Malgré qu'il n'y ait pas encore eu d'analyse de tâches qui, normalement, facilite la définition des critères d'utilisabilité, nous proposons d'en définir dès le début du projet pour faciliter le déroulement des autres activités. Les critères énoncés durant l'itération de préparation peuvent et devraient être revus à chaque itération.

(05) Construire des personas à partir des caractéristiques des utilisateurs

Afin de rendre plus concrètes les caractéristiques des utilisateurs, il est proposé de construire des personas et de les utiliser dans la rédaction de récits utilisateurs. On s'assure ainsi que l'équipe de développement vise bien les utilisateurs ciblés par le logiciel.

(06) Rédiger des scénarios de haut niveau

À partir des objectifs du système, définir des scénarios de haut niveau (pour le système). Ceux-ci sont similaires à des thèmes ou des « modules » à inclure dans le logiciel; par exemple, un logiciel de facturation est découpé en modules de clients, de factures, d'utilisateurs du logiciel, etc. Les scénarios de haut niveau n'ont pas à être exhaustifs durant l'itération de préparation puisque de nouveaux scénarios seront rédigés et détaillés durant les itérations à venir. Il suffit d'en rédiger suffisamment pour représenter ce que nous connaissons au moment de l'itération de préparation du système.

(07) Concevoir un schéma de navigation à partir des scénarios de haut niveau

Concevoir un schéma permettant de voir et de comprendre comment les différents scénarios de haut niveau sont interreliés. Ceci donne à toute l'équipe une vision sur les différents modules et sur les liens potentiels entre eux ou permet de clarifier leur indépendance. Dans cette activité, le schéma de navigation ne traite pas spécifiquement des différents menus qui seront présents dans les interfaces utilisateurs, ni des détails de la séquence des interfaces. Il s'agit bien de comprendre les relations entre les modules présentés dans les scénarios de haut niveau.

(08) Rédiger des récits utilisateurs détaillés

Les récits utilisateurs sont rédigés à partir des scénarios de haut niveau et seront les éléments qui seront inclus dans le travail d'une itération.

(09) Prioriser les récits utilisateurs selon la valeur d'affaire

Le PP ordonne les récits utilisateurs en fonction de leur valeur d'affaire et de leur l'utilité pour les utilisateurs. Les membres de l'équipe travaillent de pair avec le PP pour qu'il considère les récits qui ajoutent de l'utilité au logiciel pour l'utilisateur, même s'ils semblent présenter une plus petite valeur d'affaire.

(10) Effectuer une analyse de tâche des utilisateurs pour les prochains récits utilisateurs du carnet de produits

Une analyse de tâche permet de décrire et de comprendre le travail réel de l'utilisateur et de définir le modèle de la tâche. Il est probable que le système impose une nouvelle manière de réaliser le travail. L'analyse de la tâche actuelle demeure utile pour prendre en considération toutes les informations nécessaires à l'utilisateur. Un récit utilisateur peut comprendre plusieurs tâches à effectuer qui correspondent aux différents moments de l'activité de l'utilisateur et qui déterminent en grande partie les comportements de l'utilisateur.

Dans ce guide, l'analyse de tâche est réalisée après la définition des récits utilisateurs et non au début comme on s'y attend dans une approche de CCU. Ce choix est volontairement fait pour réduire la portée de l'analyse de tâche à ce que le PP tente de réaliser dans le logiciel. Ainsi, il est possible de réaliser l'analyse de tâche concernant le récit traité dans une itération et en faire ressortir les informations utiles et nécessaires à la conception et au développement du récit correspondant.

(11) Décrire les lignes directrices et les standards de conception d'interfaces

Les lignes directrices de conception d'interfaces sont de précieuses références pour l'équipe Scrum. Elles portent sur tout un ensemble d'éléments de l'interface (ex., terminologie, navigation, guidage, disposition graphique, erreurs, couleurs, etc.) et décrivent le comportement des diverses composantes de l'interface (ex., menus, fenêtres, retour d'information, messages, etc.). Ces éléments débutent parfois par des contraintes ou des lignes directrices énoncées par l'organisation.

(12) Concevoir des maquettes d'interfaces utilisateurs pour les récits utilisateurs analysés

La conception de maquettes devrait être faite par tous les membres de l'équipe Scrum. L'ergonome de l'équipe s'assure continuellement de la prise en compte des caractéristiques des utilisateurs. Le développeur s'assure que les concepts sont techniquement réalisables, en considérant les capacités et les limites des technologies utilisées dans la réalisation du logiciel

(13) Valider les maquettes conçues, auprès d'utilisateurs

La validation de maquettes d'interface fait ressortir des corrections ou des améliorations à faire. Celles-ci peuvent être rapidement apportées aux maquettes, avant même le début du développement du récit.

(14) Estimer le travail de réalisation des récits utilisateurs

L'estimation du travail de réalisation des récits utilisateurs par l'équipe permet au PP de connaître le coût de développement des récits qu'il rédige. Elle est un pré-requis à la planification des itérations puisqu'elle permet de connaître avec précision la quantité de travail réalisable durant cette itération.

(15) Planifier l'itération à partir des éléments du carnet de produit

Durant la séance de planification d'itération, l'équipe s'engage avec le PP à réaliser une certaine quantité de travail durant l'itération qui débute. Le travail à réaliser concerne autant la conception de récits utilisateurs par les ergonomes que le développement par les développeurs des récits préalablement conçus par l'équipe.

(16) Réaliser le développement des éléments planifiés pour l'itération

À cette étape, l'équipe travaille sur la conception et la réalisation technique des récits utilisateurs qui ont été planifiés lors de la séance de planification de l'itération courante.

(17) Présenter au propriétaire du produit le travail réalisé durant l'itération

À la fin de chaque itération, l'équipe présente au PP le travail réalisé durant l'itération. La présentation au PP est la confirmation par l'équipe que le travail réalisé durant l'itération correspond effectivement aux attentes énoncées dans les récits utilisateurs et à l'engagement pris durant la séance de planification.

Cette présentation peut être considérée comme un test d'utilisabilité préliminaire où le PP agit en tant qu'utilisateur. C'est lors de l'itération suivante que des utilisateurs supplémentaires seront amenés à faire des validations. Les comportements du PP ne doivent pas être considérés comme représentatifs du comportement typique d'un utilisateur du système. Par contre, ils permettent d'obtenir un premier ensemble de données.

(18) Valider le logiciel réalisé auprès d'utilisateurs

Avant la mise en production du logiciel, la validation auprès d'utilisateur est un moyen très efficace pour évaluer ce logiciel et y apporter des modifications, si nécessaire. La validation est effectuée durant une séance de tests d'utilisabilité où l'équipe observe des utilisateurs face au logiciel. Cette méthode permet de relever les problèmes que rencontre l'utilisateur avec le logiciel.

Les tests d'utilisabilité permettent de connaître le comportement des utilisateurs et de vérifier qu'il correspond à celui attendu. Lorsque nécessaire, l'équipe suggère des modifications au mode d'utilisation du logiciel afin qu'il corresponde mieux aux attentes des utilisateurs.

La validation par tests d'utilisabilité à chaque itération peut sembler beaucoup de travail, mais les tests sont effectués sur la partie du logiciel qui concerne le récit utilisateur qui a été développé, et non pas un scénario complet. Les corrections/modifications trouvées créent de nouveaux éléments dans le carnet de produit et le PP doit les prioriser, de la même manière qu'il le fait avec les récits utilisateurs.

4.1.2 Activités de l'itération de préparation

Les activités de l'itération de préparation sont les suivantes :

(01) Prendre connaissance du projet, de la vision du propriétaire du produit et des objectifs du logiciel;
(02) Analyser le contexte actuel;
(03) Définir les utilisateurs visés et leurs caractéristiques;
(04) Établir les critères/objectifs d'utilisabilité;
(05) Construire des personas à partir des caractéristiques des utilisateurs;
(06) Rédiger des scénarios de haut niveau;
(07) Concevoir un schéma de navigation à partir des scénarios de haut niveau;
(08) Rédiger des récits utilisateurs détaillés;
(09) Prioriser les récits utilisateurs selon la valeur d'affaire;

(10) Effectuer une analyse de tâche des utilisateurs pour les prochains récits utilisateurs du carnet de produit;

(11) Décrire les lignes directrices et les standards de conception d'interfaces;

(12) Concevoir des maquettes d'interfaces utilisateurs pour les récits utilisateurs analysés;

(13) Valider les maquettes conçues auprès d'utilisateurs;

(14) Estimer le travail de réalisation des récits utilisateurs.

4.1.3 Activités de l'itération 1

Les activités de l'itération 1 sont les suivantes :

(15) Planifier l'itération à partir des éléments du carnet de produit;

(16) Réaliser le développement des éléments planifiés pour l'itération;

(10) Effectuer une analyse de tâche des utilisateurs pour les prochains récits utilisateurs du carnet de produit;

(12) Concevoir des maquettes d'interfaces utilisateurs pour les récits utilisateurs analysés;

(13) Valider les maquettes conçues auprès d'utilisateurs;

(17) Présenter au propriétaire du produit le travail réalisé durant l'itération.

4.1.4 Activités des itérations 2 à n

Les activités des itérations 2 à n sont les suivantes :

(14) Estimer le travail de réalisation des récits utilisateurs;

(15) Planifier l'itération à partir des éléments du carnet de produit;

(16) Réaliser le développement des éléments planifiés pour l'itération;

(18) Valider auprès d'utilisateurs le logiciel réalisé;

(10) Effectuer une analyse de tâche des utilisateurs pour les prochains récits utilisateurs du carnet de produit;

(12) Concevoir des maquettes d'interfaces utilisateurs pour les récits utilisateurs analysés;

(13) Valider les maquettes conçues auprès d'utilisateurs;

(17) Présenter au propriétaire du produit le travail réalisé durant l'itération.

4.2 Évaluation du guide d'intégration

L'évaluation du guide se déroule en deux phases. Les résultats présentés dans ce travail proviennent de la première phase de validation. La deuxième phase étant prévue se poursuivre après la remise du mémoire, aucun résultat ne peut être présenté. Avant de débuter la première phase d'évaluation, le projet de recherche a été soumis au Comité d'éthique de la recherche (CÉR). Le certificat d'acceptation du projet de recherche est présenté en annexe 2.

4.2.1 Première phase d'évaluation

L'objectif de cette phase d'évaluation est de confronter le guide aux principes et valeurs guidant les pratiques de chaque domaine. Ceci permet de vérifier si, selon les évaluateurs, les activités présentées dans le guide respectent les principes et valeurs, puis sont applicables et réalisables dans le contexte d'un projet de développement logiciel, si elles permettent de concevoir et développer un logiciel utile et utilisable pour ses utilisateurs, et enfin, que le format de présentation et le contenu du guide offrent un outil pratique pour les équipes qui l'utilise.

Pour effectuer l'évaluation du guide, 11 personnes ont été rencontrées : six spécialistes du développement agile et cinq spécialistes de la CCU. La description des participants est détaillée dans le tableau 3. Dans chaque groupe, au moins deux sujets avaient des connaissances égales des deux domaines.

Les 11 participants qui ont été recrutés comme évaluateur du guide d'intégration sont des professionnels que nous connaissons et que nous avons contactés par courriel. Le courriel de recrutement expliquait aux participants les détails du projet et de leur participation. L'information présentée dans le courriel est la suivante :

- Objectif du travail de recherche : proposer par l'entremise d'un guide, une méthode qui permet d'« Intégrer une approche de conception centrée utilisateur à une approche agile de développement logiciel »;

- Objectif de leur participation : être évaluateur du guide d'intégration proposé;
- Travail à réaliser : lire le guide d'intégration et y inscrire, aux endroits appropriés, des commentaires et suggestions d'améliorations, s'il y a lieu. Par la suite, répondre à des questions durant une entrevue semi-dirigée téléphonique ou en personne, afin de partager les commentaires et suggestions d'améliorations avec l'auteur du guide. Durant l'entrevue, les commentaires sont recueillis par écrit et un enregistrement vocal est fait en parallèle.
- Durée : environ 1 heure d'entrevue
- Compensation : les participants reçoivent une compensation monétaire de 25$.

Tableau 3: Description des caractéristiques des évaluateurs du guide

Évaluateur	**Âge, Sexe**	**Occupation**	**Domaine d'expertise**	**Années d'expérience**
1	35-40, M	Propriétaire du produit, accompagnateur d'équipe agile	Agile	8
2	25-30, M	Développeur	Agile	2
3	45-50, M	Ergonome, professeur	CCU	8
4	30-35, M	Ergonome	CCU	3
5	35-40, M	Ergonome	CCU	6
6	25-30, M	Développeur	Agile	3
7	25-30, M	Développeur	Agile	2
8	40-45, M	Propriétaire du produit, accompagnateur d'équipe agile	Agile	3.5
9	30-35, M	Analyste-développeur, chercheur	CCU	6
10	25-30, M	Développeur	Agile	2
11	45-50, M	Ergonome, professeur	CCU	20

4.2.2 Deuxième phase d'évaluation

L'objectif de la deuxième phase d'évaluation est de vérifier l'application et l'utilisation du guide dans des contextes réels de projet de développement logiciel. Pour la deuxième phase d'évaluation, des équipes de développement logiciel travaillant selon une approche Scrum et qui utilisent le guide seront suivies. Des rencontres auront lieu pour recueillir les commentaires et suggestions des équipes. Les commentaires devraient, entre autres, être orientés sur l'utilité des activités du guide, la séquence proposée, les ajustements nécessaires à la durée ou la méthode de réalisation suggérée pour les activités et le format de présentation des éléments du guide.

4.3 Résultats de l'évaluation

Les nombreux commentaires recueillis portent sur le rôle, le contenu et la présentation du guide. La principale différence entre les commentaires des deux groupes de sujets tient à la plus forte présence d'éléments relevant de leur discipline d'appartenance (p. ex., les aspects techniques pour les développeurs agiles). Les commentaires sont présentés sans faire de distinction entre les deux groupes et sans identifier l'évaluateur puisque certains des commentaires ont été faits par plusieurs sujets durant les entrevues.

De façon générale, on souligne que le guide porte sur une problématique très pertinente et qu'il peut être fort utile pour le développement de logiciel. Il est aussi vu comme ambitieux et sa mise en oeuvre représente un défi puisque le travail de CCU peut devenir un goulot d'étranglement si les activités qui lui sont associées ne sont pas bien réparties dans l'équipe. Dans le texte qui suit, nous rapportons les principaux commentaires et suggestions des évaluateurs, puis nous discutons de ces commentaires pour mieux comprendre comment ils seront traités pour l'amélioration du guide.

Commentaire 1 : (Observation) Les rôles du ScrumMaster ou celui du concepteur graphique ne sont pas mentionnés dans le guide.

Réponse : (Cette observation n'a pas mené à une modification du guide). Ces rôles sont importants pour la réalisation d'un logiciel selon une approche agile. Cependant, le rôle du ScrumMaster ne change pas avec l'ajout d'activités de CCU à Scrum. Il doit continuer à être le gardien du processus, à s'assurer que l'équipe progresse au rythme prévu et à enlever les obstacles qui nuisent à l'équipe. Pour ce qui est du rôle du concepteur graphique, il n'est pas traité explicitement, tout comme les autres rôles spécialisés (p. ex. : administrateur de base de données, architecte système, spécialiste de l'assurance qualité, etc.). Deux raisons expliquent cette omission : la première est que les spécialistes ne font pas partie de l'équipe Scrum et sont consultés au besoin, ce qui fait qu'ils n'ont pas à être mentionnés dans ce guide. La deuxième raison est que s'ils font partie de l'équipe Scrum, leur spécialisation est déjà considérée dans la planification des itérations et du travail de l'équipe.

Commentaire 2 : (Observation) La présentation de la séquence d'activités dans un tableau n'est pas assez visuelle et ne permet pas de bien suivre la séquence d'activités à réaliser, ni de bien préparer les activités suivantes.

Réponse : (Cette observation a mené à une modification du guide). Le tableau initialement utilisé est remplacé par un schéma représentant les itérations de Scrum auxquelles les activités de CCU sont ajoutées. De plus, le guide ne permet pas de bien suivre un élément à travers les itérations; un élément du carnet de produit est décrit à une itération, des analyses de tâches et maquettes sont faites pour détailler le contenu à l'itération suivante, puis le développement suit et enfin, des tests d'utilisabilité sont effectués. Décrire cette séquence dans un schéma est plus efficace que dans un tableau.

Commentaire 3 : (Observation) Le guide ne fait pas de distinction entre les activités Scrum et celle d'ergonomie.

Réponse : (Cette observation n'a pas mené à une modification du guide). L'objectif premier du guide est de proposer une intégration des activités de CCU à celles d'une approche agile, alors faire une distinction semble aller à l'opposé de cet objectif. De plus, une équipe Scrum est considérée comme multidisciplinaire et tous ses membres ont les connaissances pour réaliser l'engagement de l'itération. Le fait de distinguer les activités de chaque domaine contribue à

conserver un fossé entre les spécialistes agiles et les spécialistes de la CCU plutôt que d'adhérer au principe de Scrum.

Commentaire 4 : (Suggestion) En principe, les ergonomes font une partie du travail en amont des développeurs. Ce qui n'est pas clairement exprimé dans le guide. Il serait pertinent de préciser qu'il y a un décalage de travail entre les ergonomes et les développeurs.

Réponse : (Cette suggestion a mené à l'ajout des précisions dans le guide). Le guide ne spécifie pas clairement le travail des ergonomes, puisqu'il considère que le travail est réalisé par les membres de l'équipe et non spécifiquement par un ergonome ou un développeur. Par contre, il y a un certain décalage entre le travail de CCU effectué sur un récit utilisateur et la réalisation technique du récit. Ceci pour permettre à l'équipe « d'analyser » le récit, du point de vue utilisateur, avant de débuter le développement.

Commentaire 5 : (Observation) Le guide ne fait pas mention des aspects de génie logiciel (intégration continue, réusinage, etc.).

Réponse : (Cette observation n'a pas mené à une modification du guide). L'objectif du guide est de présenter une séquence d'activités de CCU qui s'intègre à celles de Scrum. Les pratiques de génie logiciel qui sont déjà en place dans les équipes continuent d'exister et d'être mises en pratique. Ces aspects ne sont pas traités dans le guide puisqu'il deviendrait trop volumineux de mentionner toutes les autres pratiques des équipes Scrum. Il parait plus efficace de concentrer le contenu sur les activités de CCU.

Commentaire 6 : (Observation) Le niveau d'importance n'aide pas vraiment au choix d'activités à réaliser selon le contexte du projet.

Réponse : (Cette observation a mené à une modification du guide). Il est adéquat de considérer toutes les activités comme importantes et devant être réalisées. Ce qui semble nécessaire est de connaître les conséquences de ne pas effectuer une certaine activité. Ainsi, les membres de l'équipe peuvent faire un choix plus éclairé sur les activités à effectuer dans le contexte de leur

projet. Pour cette raison, le niveau d'importance initialement présenté dans le guide a été retiré et remplacé par une explication des conséquences d'omettre l'activité en cours.

Commentaire 7 : (Suggestion) L'activité 10 (Effectuer une analyse de tâche des utilisateurs pour les prochains récits utilisateurs du carnet de produit) devrait être réalisée plus tôt dans le processus puisque c'est à partir des analyses de tâche qu'il est possible de définir les récits.

Réponse : (Cette suggestion n'a pas mené à une modification du guide). Dans le but de minimiser le travail à faire et de fournir un minimum d'informations utiles à l'équipe Scrum, les analyses de tâches sont réalisées après que le propriétaire du produit ait défini des récits utilisateurs. C'est à partir de l'énoncé des récits utilisateurs que les analyses de tâches sont réalisées. Ceci permet de connaître comment un utilisateur effectue son travail, mais de façon très ciblée sur un élément de travail, contrairement à une description plus globale et complète faite traditionnellement par le travail d'ergonomes.

Commentaire 8 : (Observation) Le guide ne précise pas comment gérer les commentaires/suggestions et corrections qui ressortent suite aux validations auprès d'utilisateurs.

Réponse : (Cette observation a mené à une modification du guide). Les corrections sont apportées au carnet de produit et sont gérées par le propriétaire du produit comme il le fait pour les autres éléments du carnet de produit.

Commentaire 9 : (Observation) Il est difficile de juger du niveau de détails à faire ressortir lors de la réalisation d'une activité.

Réponse : (Cette observation a mené à une modification du guide). Il s'agit de fournir un minimum de détails pour que l'équipe puisse avancer avec la conception et la réalisation du logiciel. À chaque itération, l'équipe peut s'ajuster pour détailler plus ou moins certaines activités selon les résultats obtenus lors des itérations précédentes. De plus, pour guider les équipes, un exemple de livrable attendu est ajouté au guide révisé.

Commentaire 10 : (Observation) Le guide n'identifie pas clairement quels sont les utilisateurs visés : les équipes Scrum désirant appliquer une approche CCU, les équipes Scrum comprenant un ou des ergonomes, les équipes qui n'utilisent pas une approche Scrum et qui veulent débuter, etc.

Réponse : (Cette observation a mené à une modification du guide). Le guide vise d'abord les équipes Scrum désirant appliquer une approche CCU puisqu'il est bâti en ajoutant aux activités de Scrum un certain nombre d'activités provenant du domaine de la CCU. Cependant, le guide peut être bénéfique à toutes les équipes de développement logiciel parce qu'il est susceptible d'aider aussi ceux qui n'ont pas de ressources en ergonomie.

Commentaire 11 : (Suggestion) Le guide ne précise pas le nombre d'ergonomes à intégrer à l'équipe Scrum. Deux ergonomes devraient participer au projet.

Réponse : (Cette suggestion n'a pas mené à une modification du guide). La composition de l'équipe ne relève pas de ce guide. Quoiqu'il soit nécessaire de garder un équilibre pour s'assurer de représenter chaque « spécialisation de métier » au sein de l'équipe, il est difficile de proposer un nombre précis de personnes. Il est mentionné [3] qu'une équipe Scrum est composée d'environ sept personnes, plus ou moins deux, alors la suggestion d'intégrer deux ergonomes à l'équipe semble pertinente pour éviter que le travail de CCU devienne un goulot d'étranglement.

Commentaire 12 : (Observation) Le papier n'est pas nécessairement le plus efficace pour la conception des maquettes.

Réponse : (Cette observation n'a pas mené à une modification du guide). Nous sommes d'accord avec cette observation. Dans certaines situations, il est difficile d'exprimer clairement le comportement d'une interface par l'entremise d'une maquette papier. Cependant, le guide tente de mettre en place des activités qui répondent à la majorité des situations et laisse aux équipes Scrum le soin d'apporter des solutions spécifiques aux situations non abordées par le guide. Ainsi, le guide n'est pas modifié suite à cette observation, mais pourrait l'être ultérieurement si une solution pratique est proposée par les équipes qui utilisent le guide.

Commentaire 13 : (Observation) Le principe d'itération finale n'est pas clair : dans une approche Scrum n'importe quelle itération peut être la dernière.

Réponse : (Cette observation a mené à une modification du guide). La version initiale comportait des activités pour l'itération finale, celle menant réellement à une mise en production du logiciel. Pour faire suite à cette observation, nous avons retiré l'itération finale durant la révision du guide. Le retrait de cette dernière reflète plus la réalité rencontrée durant nos expériences de projets de développement selon une approche Scrum, alors que la version initiale tentait de s'approcher des pratiques de CCU où un projet a une fin claire et définie dès le début.

Commentaire 14 : (Suggestion) Il serait utile d'orienter le lecteur vers des références externes ou de donner des exemples de méthodes pour réaliser les activités.

Réponse : (Cette suggestion a mené à l'ajout des précisions dans le guide). En plus de suggérer une méthode pour réaliser l'activité, la révision du guide ajoute un exemple ou une référence vers une méthode connue. Ceci pour aider les équipes qui sont moins familières avec ce qui est mentionné dans le guide.

Commentaire 15 : (Observation) Il ne semble pas évident de faire des évaluations auprès des utilisateurs à la fréquence mentionnée. Peut-elle être diminuée sans nuire à la qualité ?

Réponse : (Cette observation n'a pas mené à une modification du guide). L'observation est pertinente. Cependant, il n'est pas possible d'y donner suite avec les données actuellement connues. Il sera possible de le faire après avoir obtenu des commentaires par les équipes qui utilisent le guide.

Commentaire 16 : (Suggestion) Il serait plus utile de mentionner qu'il y a une révision continue des artéfacts lors des activités, plutôt que d'en faire une activité spécifique.

Réponse : (Cette suggestion a mené à l'ajout des précisions dans le guide). La première version du guide incluait une activité « Réviser les lignes directrices et les standards de conception d'interfaces ». Cette activité a été retirée suite aux suggestions reçues. En effet, il n'est pas

nécessaire de préciser une activité de révision, puisque dans le cycle itératif de Scrum, il est possible à chaque itération de revoir et d'améliorer le processus et les pratiques en place.

Commentaire 17 : (Observation) Il peut être difficile de mettre en place au début du projet, lors de l'itération de préparation, les critères d'utilisabilité avec le PP.

Réponse : (Cette observation n'a pas mené à une modification du guide). La mise en place des critères d'utilisabilité semble importante dès le début du projet pour bien guider la conception par la suite. Par contre, l'utilité de les définir si tôt pourrait être remise en question ultérieurement selon les commentaires obtenus par les équipes Scrum.

Commentaire 18 : (Suggestion) Il est difficile d'évaluer l'implication des différents intervenants Il serait utile de préciser le niveau d'implication des intervenants selon l'itération du projet.

Réponse : (Cette suggestion n'a pas mené à une modification du guide). Cette suggestion n'a pas été retenue puisqu'elle va à l'encontre d'un objectif du guide : réunir les ergonomes et les développeurs agiles au sein d'une équipe ayant un engagement commun. Ainsi, tous les membres de l'équipe sont entièrement impliqués dans le projet et ce, tout au long du projet.

Commentaire 19 : (Suggestion) Pour bien prendre en considération l'effort associé au travail de CCU, l'ergonome devrait être présent aux séances d'estimation (activité #14).
Réponse : (Cette suggestion n'a pas mené à une modification du guide). La suggestion est pertinente mais ne demande pas de modifications au guide. L'ergonome est présent dans l'équipe pour toutes les activités mentionnées dans le guide, même si celles-ci proviennent du domaine du développement agile. Ceci demeure vrai pour les développeurs qui sont présents lors de la réalisation des activités provenant du domaine de la CCU. Le guide propose une méthode qui tente d'amener tous les intervenants à agir comme des membres d'une seule équipe.

Commentaire 20 : (Observation) Le guide ne peut pas être pris tel quel. Il nécessite un minimum de connaissances des activités de l'un ou l'autre des domaines.

Réponse : (Cette observation n'a pas mené à une modification du guide). Il est difficile de savoir quel niveau de connaissance est requis pour utiliser le guide tel qu'il est présenté. Nous estimons

qu'il sera possible d'en connaître plus lors des évaluations ultérieures. Cependant, comme l'équipe devrait inclure des développeurs agiles et des ergonomes, nous croyons que les connaissances pourront être partagées au sein de l'équipe.

Commentaire 21 : (Observation) Les détails de l'activité d'analyse de tâche ne sont pas clairs.

Réponse : (Cette observation a mené à une modification du guide). Les détails de l'activité d'analyse de tâche ont été revus et étoffés autant que possible sans alourdir le guide lors des corrections menant à la deuxième version du guide.

Autres commentaires ne nécessitant pas de suivi :

- Très bien que l'ergonome fasse partie de l'équipe et qu'il soit présent lors de l'activité #17 « Présenter au propriétaire du produit le travail réalisé durant l'itération ».
- Le guide présente une bonne intégration parce qu'il y a des interfaces de travail entre les « spécialistes » et aussi avec le PP.
- La méthode proposée est bonne, elle reste ouverte. Le guide apporte une séquence d'activités sans trop restreindre la portée de chacune et laisse de la flexibilité aux équipes.
- Pour bien intégrer les concepts au sein des équipes, le guide devrait être intégré à un guide Scrum.
- Il y a une bonne couverture des activités [provenant de chaque domaine].

4.4 Améliorations apportées au guide

Suite aux commentaires et aux suggestions d'améliorations obtenues durant la première phase de validation, certaines améliorations ont été apportées au guide. Nous détaillons maintenant ces améliorations.

Un résumé du guide est maintenant présenté sous forme de schéma et non plus sous forme de tableau d'activités. Ce schéma sert d'abord de résumé du guide, et une fois le contenu connu assimilé par les membres des équipes, il peut être utilisé seul comme simple référentiel tout au long du projet.

Les informations présentées pour détailler chaque activité n'étaient pas jugées complètement utiles, alors elles ont été modifiées pour tenir compte des suggestions des sujets. La version initiale du guide présentait des informations sur chacun des points suivants :

- Intervenants : les intervenants mentionnés sont recommandés, alors que les autres sont facultatifs.
- Calendrier : l'itération où devrait se dérouler l'activité.
- Description : des détails sur l'activité et les raisons de la réaliser.
- Importance : Haute, moyenne ou basse; une échelle à trois niveaux permet aux équipes de mieux décider si l'activité doit être réalisée dans le contexte du projet.
- Livrable : ce que les intervenants devraient produire durant l'activité.
- Temps : une période de temps suggérée pour réaliser l'activité.
- Méthode utilisée : une méthode suggérée pour réaliser l'activité.

Pour combler les lacunes mentionnées par les sujets, la version révisée du guide contient maintenant les informations suivantes :

- Description des détails sur l'activité et les raisons de la réaliser.
- Conséquences de ne pas réaliser l'activité. Cette information était jugée plus pratique par des évaluateurs du guide.
- Intervenants : les intervenants mentionnés sont recommandés alors que les autres sont facultatifs.
- Calendrier : l'itération où devrait se dérouler l'activité.

- Livrable : ce que les intervenants devraient produire durant l'activité.
- Exemple : un livrable que l'équipe devrait produire. Ceci pour faciliter le travail des équipes qui ne connaissent pas certaines des activités mentionnées dans le guide.
- Temps alloué : une période de temps suggérée pour réaliser l'activité.
- Méthode allégée : une méthode suggérée pour réaliser l'activité en y faisant ressortir un minimum d'informations utiles.
- Méthode plus élaborée : une méthode suggérée pour réaliser l'activité de manière plus complète pour aller chercher le maximum d'informations pour faire avancer la conception.

CONCLUSION

Nous avons élaboré et évalué un guide de travail présentant une approche intégrée de développement logiciel qui mise sur les avantages des approches ergonomique et agile. Les éléments présentés ont un grand potentiel pour permettre de bien intégrer les pratiques des deux domaines. De plus, l'évaluation faite indique qu'un tel guide contribue à créer un logiciel qui apporte de la valeur, autant pour le propriétaire du produit que pour l'utilisateur final. La méthode d'intégration proposée dans le guide est avant tout un outil pour les équipes de développement. Cependant, cette méthode d'intégration n'est pas suffisante à elle seule. Pour réussir, elle doit être accompagnée d'un changement de mentalité pour tous les intervenants à la conception et au développement de logiciel. Ces intervenants sont autant les ergonomes, l'équipe de développement que le propriétaire du produit. Chacun doit avoir le courage d'accepter de modifier ses pratiques : le propriétaire du produit doit apprendre à voir et valoriser des incréments de logiciel axés sur la simplification de l'interface du système et la mise en valeur des utilisateurs; les équipes de développement doivent accepter de travailler sur des concepts du logiciel avant même de pouvoir commencer la programmation; les spécialistes en ergonomie doivent apprendre à participer à la conception d'un logiciel par petits incréments en réalisant au moins un minimum d'activités utiles, et ce, au moment le plus opportun dans le cycle de vie du logiciel.

Le guide propose un modèle pertinent et utile pour les équipes de développement logiciel. Cependant, les travaux qui ont menés à la conception de ce modèle ont quelques limitations. D'abord, l'évaluation du guide proposé a été réalisée auprès d'un nombre restreint (11), mais jugé suffisant, d'évaluateurs. Par ailleurs, certains des évaluateurs possèdent une courte expérience dans le développement logiciel agile (entre 2 et 3 années). Cela peut amener un biais dans l'évaluation puisque les développeurs agiles n'avaient pas nécessairement le bagage nécessaire pour critiquer le modèle et faire ressortir des éléments d'amélioration qui le rendrait plus utile. Également, cette expérience limitée a pu affecter la capacité des évaluateurs à mettre en lumière des éléments du guide qui pourraient nuire à l'efficacité du déroulement d'une itération de réalisation du logiciel ou à situer le guide dans différents projets de développement logiciel.

Une autre limite du projet est que le modèle proposé se veut général et tente de répondre à une multitude de contextes de projets de développement logiciel. Bien que cela facilite la généralisation de la méthode, cette caractéristique pourrait complexifier l'utilisation du guide puisqu'il est difficile d'identifier précisément le type de projets visés (le domaine d'application, l'envergure du projet, les délais requis, etc.). Ainsi, il reste à déterminer les contextes où l'approche proposée sera plus simple à mettre en pratique. Dans sa version actuelle, notre approche est jugée plus adaptée pour les logiciels de gestion ou de commerce électronique créés pour un client spécifique puisque des utilisateurs réels peuvent facilement être mis au premier plan dans la conception.

Pour être utile aux équipes et pour mitiger l'impact des limitations mentionnées, le guide présenté doit continuer d'évoluer pour satisfaire aux exigences réelles de conception et de développement de logiciels. Afin de s'assurer qu'il évolue et répond aux besoins des équipes visées par le guide, il est nécessaire de compléter son évaluation et de réviser son contenu pour s'ajuster aux résultats de ces évaluations. Entre autres, nous devons faire appel à un plus grand nombre de sujets, qui possèdent une plus grande expérience dans le développement agile ou la conception centrée utilisateur dans divers projets. À travers ces évaluations, nous pourrons vérifier si le guide est vraiment utilisé, quelles informations sont vraiment utilisées par les équipes, quelle partie du guide a été personnalisée par les équipes, etc. Enfin, ces informations pratiques sur l'utilisation du guide nous permettront d'évaluer son impact sur le travail des développeurs et des ergonomes durant le processus de conception et de développement d'un logiciel et d'apporter les modifications nécessaires pour augmenter son utilité.

Nous sommes d'avis que ce travail se situe dans la continuité du rapprochement qui est déjà amorcé entre les communautés prônant ces approches. L'élaboration du guide d'intégration a été faite avec une méthode qui se rapproche de celles utilisées dans le cadre des travaux d'autres auteurs. D'abord, tout comme [41], ce travail de recherche est, entre autre, basé sur notre expérience de travail au sein d'équipe travaillant avec une approche agile. Cette expérience permet de proposer un guide utile et traitant le sujet de manière pragmatique. Cette recherche a

fait appel à notre expérience professionnelle ainsi qu'aux expériences et connaissances de spécialistes de chaque domaine qui ont, pour certains, évolués autant dans un contexte Scrum que de CCU. Cette méthodologie de recherche, aussi utilisé par [42], [43] et [44], est utile pour enrichir notre propre expérience avec celles d'autres spécialistes et permet d'avoir un modèle plus réaliste. Ces différentes expériences ont permis de rendre la méthode proposée pratique et applicable. Néanmoins, ce travail de recherche origine également de divers travaux existants pour s'assurer d'une base solide et garantir que la méthode proposée ici soit une contribution significative aux écrits existants. Dans ce contexte, il a été nécessaire pour débuter de critiquer, faire ressortir des défis et décrire les approches déjà étudiées, comme l'a fait [45]. Enfin, ces orientations de recherche ont permis de proposer une méthode de travail qui tente de combler des lacunes de Scrum dans ses considérations orientées utilisateur [46], sans toutefois perdre de vue les principes agiles qui guident cette approche.

Il reste beaucoup à faire pour faciliter l'adoption et la mise en oeuvre d'une approche intégrée. Chaque participant à la réalisation du logiciel doit comprendre et assimiler que chacun fait partie intégrante d'une seule équipe ayant un objectif commun et qu'il développe un logiciel qui doit à la fois être utile, utilisable et procurer une expérience utilisateur positive pour ses utilisateurs. Ceci ne peut se réaliser qu'avec la contribution de chacun des membres d'une équipe avec ses connaissances, son expérience et son expertise et sa pleine intégration au projet.

RÉFÉRENCES

[1] J.-F. Proulx et J.-M. Robert, « Intégrer une approche de conception centrée utilisateur à une approche agile de développement logiciel » in *Actes du colloque IHM 2010 (22ème conférence francophone sur les interactions Homme-machine)*, Luxembourg, Luxembourg, 2010, pp. 125-128.

[2] P. Heller, « A Developer's Confessions About User Experience », *Usable Apps blog*, [En ligne]. Disponible : http://blogs.oracle.com/usableapps/2008/01/a-developers-confessions-about.html. [Consulté le 23 juillet 2009].

[3] K. Schwaber et J Sutherland, « Scrum guide », *Scrum.org: The Home of Scrum And Your Source For Scrum Training*, [En ligne]. Disponible : http://www.scrum.org/scrumguides/. [Consulté le 17 mars 2010].

[4] L. Miller et D. Sy, « Agile User Experience SIG » in *Proceedings of the 27th international conference extended abstracts on Human factors in computing systems*, Boston, États-Unis, 2009, pp. 2751-2754.

[5] S.W. Ambler, « Disciplined Agile Software Development: Definition », *Disciplined Agile Software Development: Definition*, [En ligne]. Disponible : http://www.agilemodeling.com/essays/agileSoftwareDevelopment.htm. [Consulté le 28 juillet 2009].

[6] K. Beck et al., « Manifesto for Agile Software Development », *Manifesto for Agile Software Development*, [En ligne]. Disponible : http://www.agilemanifesto.org/. [Consulté le 23 juillet 2009].

[7] K. Beck et al., « Principles behind the Agile Manifesto », *Principles behind the Agile Manifesto,* [En ligne]. Disponible : http://www.agilemanifesto.org/principles.html. [Consulté le 23 juillet 2009].

[8] T. Balbous, « Les méthodes Agiles - Introduction », *Les méthodes Agiles – Introduction*, [En ligne]. Disponible : http://www.slideshare.net/trems/les-mthodes-agiles-introduction-presentation [Consulté 20 août 2009].

[9] Wikipedia, « Manifeste agile », *Manifeste agile*, [En ligne]. Disponible : http://fr.wikipedia.org/wiki/Manifeste_agile. [Consulté le 20 août 2009].

[10] S. Elatta, « Destination: AgileTop Eight Reasons Why Organizations Are Making the Switch », *Scrum Alliance -Destination: AgileTop Eight Reasons Why Organizations Are Making the Switch*, [En ligne]. Disponible : http://www.scrumalliance.org/articles/84-destination-agile. [Consulté le 25 juillet 2009].

[11] auteur non mentionné. « Has Scrum Become the Face of Agile? », *Has Scrum Become the Face of Agile?*, [En ligne]. Disponible : http://scrummethodology.com/has-scrum-become-the-face-of-agile/. [Consulté le 25 juillet 2009].

[12] R. Coffin et D. Lane, « A Practical Guide to Seven Agile Methodologies, Part 1 », *A Practical Guide to Seven Agile Methodologies, Part 1,* [En ligne]. Disponible : http://www.devx.com/architect/Article/32761/0/page/2. [Consulté le 25 juillet 2009].

[13] J. Milunsky, « Agile is not just about speed », *Agile is not just about speed*, [En ligne]. Disponible : http://blog.agilebuddy.com/2009/01/agile-is-not-just-about-speed.html. [Consulté le 2 août 2009].

[14] Scrum Alliance, « Scrum Ceremonies », *Scrum Alliance -Scrum Ceremonies*, [En ligne]. Disponible : http://www.scrumalliance.org/pages/scrum_ceremonies. [Consulté le 2 août 2009].

[15] Scrum Alliance, « Scrum Artifacts», *Scrum Alliance -Scrum Artifacts*, [En ligne]. Disponible : http://www.scrumalliance.org/pages/scrum_artifacts. [Consulté le 2 août 2009].

[16] auteur non mentionné, « Manifeste Agile », *Manifeste Agile*, [En ligne]. Disponible : http://www.garance.fr/fr/methode-agile.html. [Consulté le 20 août 2009].

[17] Ergolab, « La conception centrée utilisateur », *La conception centrée utilisateur,* [En ligne]. Disponible : http://www.ergolab.net/articles/conception-centree-utilisateur.php. [Consulté le 25 juillet 2009].

[18] ISO, « Ergonomie de l'interaction homme-système – entrée sur l'opérateur humain pour les systèmes interactifs», ISO, 9241-210, 2010.

[19] ISO, « Ergonomie de l'interaction homme-système – Méthodes d'utilisabilité pour la conception centrée sur l'opérateur humain », *ISO*, TR 16982:2002, 2002.

[20] ISO, « Ergonomie des logiciels pour les interfaces utilisateur multimédias – Partie 1: Principes et cadre de conception », *ISO*, 14915-1:2002, 2002, « Partie 2: Navigation et contrôle multimédias », *ISO*, 14915-2:2003, 2003, « Partie 3: Sélection et combinaison des médias », *ISO*, 14915-3:2002, 2002.

[21] ISO, « Exigences ergonomiques pour travail de bureau avec terminaux à écrans de visualisation (TEV) – Partie 11: Lignes directrices relatives à l'utilisabilité », *ISO*, 9241-11, 1998.

[22] J.D. Gould et C. Lewis, « Designing for usability – key principles and what designers think », *Communications of the ACM*, vol. 8, no. 3, pp. 300-311, 1985.

[23] S. Chamberlain, H. Sharp et N. Maiden, « Towards a Framework for Integrating Agile Development and UCD » in *Extreme Programming and Agile Processes in Software Engineering: 7th International Conference*, Oulu, Finlande, 2006, pp. 143-153.

[24] J. Nielsen et C. Nodder, « Agile Usability: Best Practices for User Experience on Agile Development Projects », Nielsen Norman Group Report, Etats-Unis, 2008.

[25] P.N. Robillard et M. Dulipovici, « Teaching Agile Versus Disciplined Processes », *International Journal of Engineering Education*, vol. 24, no. 4, pp 671-680, 2008.

[26] D.J. Mayhew, « The Usability Engineering Lifecycle: A Practitioner's Handbook for User Interface Design », 1re édition, Morgan Kaufmann, 1999.

[27] I. Khatib, « Agile Scrum - Incorporating Usability Practices and UCD Processes in Agile Projects », *Agile Scrum - Incorporating Usability Practices and UCD Processes in Agile Projects*, [En ligne]. Disponible : http://ezinearticles.com/?Agile-Scrum---Incorporating-Usability-Practices-and-UCD-Processes-in-Agile-Projects&id=210501. [Consulté le 23 juillet 2009].

[28] W. Aigner, « Combining Extreme Programming & Interaction Design », School of Library & Information Science San Jose State University, San Jose, États-Unis, 2002, [En ligne]. Disponible : http://sunnseitn.iwolf.fastmail.fm/projects/xp_interaction_design_complete.pdf. [Consulté 01 spetembre 2009].

[29] D. Norman, « Why doing user observations first is wrong », *Don Norman's jnd.org / Why doing user observations first is wrong*, [En ligne]. Disponible : http://www.jnd.org/dn.mss/why_doing_user_obser.html. [Consulté le 27 juillet 2009].

[30] D. Sy, « Adapting Usability Investigations for Agile User-Centered Design », *Journal of Usability Studies*, vol. 2, no. 3, pp. 112-132, 2007.

[31] T. Memmel, « Agile Usability Engineering », *Agile Usability Engineering by Thomas Memmel*, [En ligne]. Disponible : http://www.interaction-design.org/encyclopedia/agile_usability_engineering.html. [Consulté le 23 juillet 2009].

[32] C. McEvoy, « Agile Usability », *Confusability: Agile Usability*, [En ligne]. Disponible : http://usability.typepad.com/confusability/2006/04/agile_usability.html. [Consulté le 23 juillet 2009].

[33] C. McEvoy, « The Software Engineering War : What does it mean for Usabilty? », *Confusability: The Software Engineering War : What does it mean for Usabilty?*, [En ligne]. Disponible : http://usability.typepad.com/confusability/2006/02/the_software_en.html. [Consulté le 27 juillet 2009].

[34] S.W. Ambler, « Tailoring Usability into Agile Software Development Projects » in *Maturing Usability: Quality in Software, Interaction and Value*, 1ère éd., E. Lai-Chong Law, E. Hvannberg, G. Cockton, Springer, 2007, pp. 75-95.

[35] J. Nielsen, « Agile User Experience Projects », *Agile User Experience Projects (Jakob Nielsen's Alertbox)*, [En ligne]. Disponible : http://www.useit.com/alertbox/agile-user-experience.html. [Consulté 4 novembre 2009].

[36] D. Churchville, « Is Agile Development Killing Usability? », *Agile Project Planning: Is Agile Development Killing Usability?*, [En ligne]. Disponible : http://www.extremeplanner.com/blog/2006/07/is-agile-development-killing-usability.html. [Consulté 16 octobre 2009].

[37] L. Miller et D. Sy, « Optimizing agile user-centred design », in *Conference on Human Factors in Computing Systems*, Florence, Italie, 2008, pp. 3897-3900.

[38] J. Patton, « Hitting the Target: Adding Interaction Design to Agile Software Development » in *Proceedings of the 17th annual ACM Conf. OOPSLA*, Seattle, Etats-Unis, pp. 1-7.

[39] Z. Hussain, W. Slany et A. Holzinger, « Curent state of agile user-centered design: A survey. », in *Proceedings of the 5th Symposium of the workgroup HCI and Usability Engineering of the Austrian computer society*, Linz, Austria, 2009, pp. 416-427.

[40] L. Schwartz, L. Vergnol, G. Gronier, A. Vagner, T. Altenburger et S. Battisti, « Comment concilier agilité et conception centrée utilisateurs dans un projet de développement? » in *Proceddings of IHM'09: 21st International Conference on Association Francophone d'Interaction Homme-Machine*, Grenoble, France, 2009, pp. 337-340.

[41] P. Hodgetts, « Experiences integrating sophisticated user experience design practices into agile processes » in *Proceedings Agile Conference 2005*, 2005, pp. 235- 242.

[42] D. Fox, J. Sillito et F. Maurer, « Agile Methods and User-Centered Design: How These Two Methodologies are Being Successfully Integrated in Industry, » in *Proceedings of Agile 2008 Conference*, Toronto, Canada, 2008, pp.63-72.

[43] Z. Hussain, W. Slany, A. Holzinger, « Investigating Agile User-Centered Design in Practice: A Grounded Theory Perspective » in *HCI and Usability for e-Inclusion, Lecture Notes in Computer Science*, vol 5889, Springer Berlin / Heidelberg, 2009, pp 279-289.

[44] J.T. Barksdale; E.D. Ragan et D.S. McCrickard, « Easing Team Politics in Agile Usability: A Concept Mapping Approach » in *Proceedings of Agile 2009 Conference*, Chicago, Illinois, 2009, pp.19-25.

[45] O. Sohaib et K. Khan, « Integrating usability engineering and agile software development: A literature review » in *2010 International Conference on Computer Design and Applications (ICCDA)*, Qinhuangdao, Chine, 2010, pp. 25-27.

[46] M. Düchting, D. Zimmermann et K. Nebe, « Incorporating User Centered Requirement Engineering into Agile Software Development » in *Interaction Design and Usability, Lecture Notes in Computer Science*, vol 4550, Springer Berlin / Heidelberg, 2007, pp 58-67.

ANNEXES

ANNEXE 1 – Guide d'intégration

GUIDE D'INTÉGRATION D'UNE APPROCHE DE CONCEPTION CENTRÉE UTILISATEUR À UNE APPROCHE AGILE (SCRUM) DE DÉVELOPPEMENT LOGICIEL

JEAN-FRANÇOIS PROULX

PYXIS TECHNOLOGIES

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

NOVEMBRE 2010

GLOSSAIRE

INTRODUCTION AU GUIDE

GUIDE

1. Prendre connaissance du projet, de la vision du propriétaire du produit et des objectifs du logiciel
2. Analyser le contexte actuel
3. Définir les utilisateurs visés et leurs caractéristiques
4. Établir les critères/objectifs d'utilisabilité
5. Construire des personas à partir des caractéristiques des utilisateurs
6. Rédiger des scénarios de haut niveau
7. Concevoir un schéma de navigation à partir des scénarios de haut niveau
8. Rédiger des récits utilisateurs détaillés
9. Prioriser les récits utilisateurs selon la valeur d'affaire
10. Effectuer une analyse de tâche des utilisateurs pour les prochains récits utilisateurs du carnet de produits
11. Décrire les lignes directrices et les standards de conception d'interfaces
12. Concevoir des maquettes d'interfaces utilisateurs pour les récits utilisateurs analysés
13. Valider les maquettes conçues, auprès d'utilisateurs
14. Estimer le travail de réalisation des récits utilisateurs
15. Planifier l'itération à partir des éléments du carnet de produit
16. Réaliser le développement des éléments planifiés pour l'itération
17. Présenter au propriétaire du produit le travail réalisé durant l'itération
18. Valider le logiciel réalisé auprès d'utilisateurs

RÉFÉRENCES

GLOSSAIRE

Carnet d'itération (sprint backlog) : Liste d'éléments du carnet de produit que l'équipe s'engage, auprès du propriétaire du produit, à réaliser durant l'itération.

Carnet de produit (product backlog) : Liste des besoins d'affaires et des fonctionnalités désirées dans le logiciel par le propriétaire du produit, généralement décrits sous forme de récits utilisateurs.

Itération (sprint) : Période de temps, entre 2 et 4 semaines, pendant laquelle l'équipe réalise les éléments du carnet de produit.

Personas : Personne fictive à laquelle on assigne un ensemble de caractéristiques et d'objectifs à réaliser, permettant de mieux représenter des utilisateurs ciblés du produit à réaliser.

Propriétaire du produit (PP) (*product owner*) : Personne responsable d'établir la vision du produit, de définir un carnet de produit et d'établir la priorité des éléments qui s'y retrouvent.

Récit utilisateur (*user story*) : Énoncé décrivant un besoin d'affaire du point de vue d'un utilisateur. Il est généralement présenté sous la forme suivante :

En tant qu'utilisateur, afin d'atteindre un objectif, je veux faire une action.

INTRODUCTION AU GUIDE

L'objectif de ce guide est de faciliter l'intégration d'activités de conception centrée utilisateur (CCU) dans un contexte agile de développement logiciel, plus précisément pour des équipes suivant une approche Scrum. Il est proposé d'ajouter aux cérémonies de Scrum certaines activités plus spécifiques à la CCU. Elles apportent des informations utiles à la réalisation du logiciel et sont réalisables dans un court laps de temps. Les activités proposées provenant d'une approche ergonomique apportent des connaissances sur les utilisateurs et leurs objectifs, afin de permettre au propriétaire du produit (PP) d'obtenir non seulement un logiciel de valeur, mais un logiciel utile et respectant les attentes des utilisateurs.

Dans ce guide, les ergonomes sont considérés comme des spécialistes faisant partie de l'équipe Scrum et non pas externes à l'équipe. Ceci correspond au principe Scrum d'avoir une équipe pluridisciplinaire dont les membres ont le même niveau d'engagement envers le développement du logiciel et possèdent les compétences nécessaires pour réaliser le produit désiré par le PP. De plus, nous avons omis de distinguer les activités de CCU et de Scrum afin de favoriser la pleine intégration des pratiques de travail relevant des deux domaines.

Les pages qui suivent décrivent les activités. Elles contiennent :

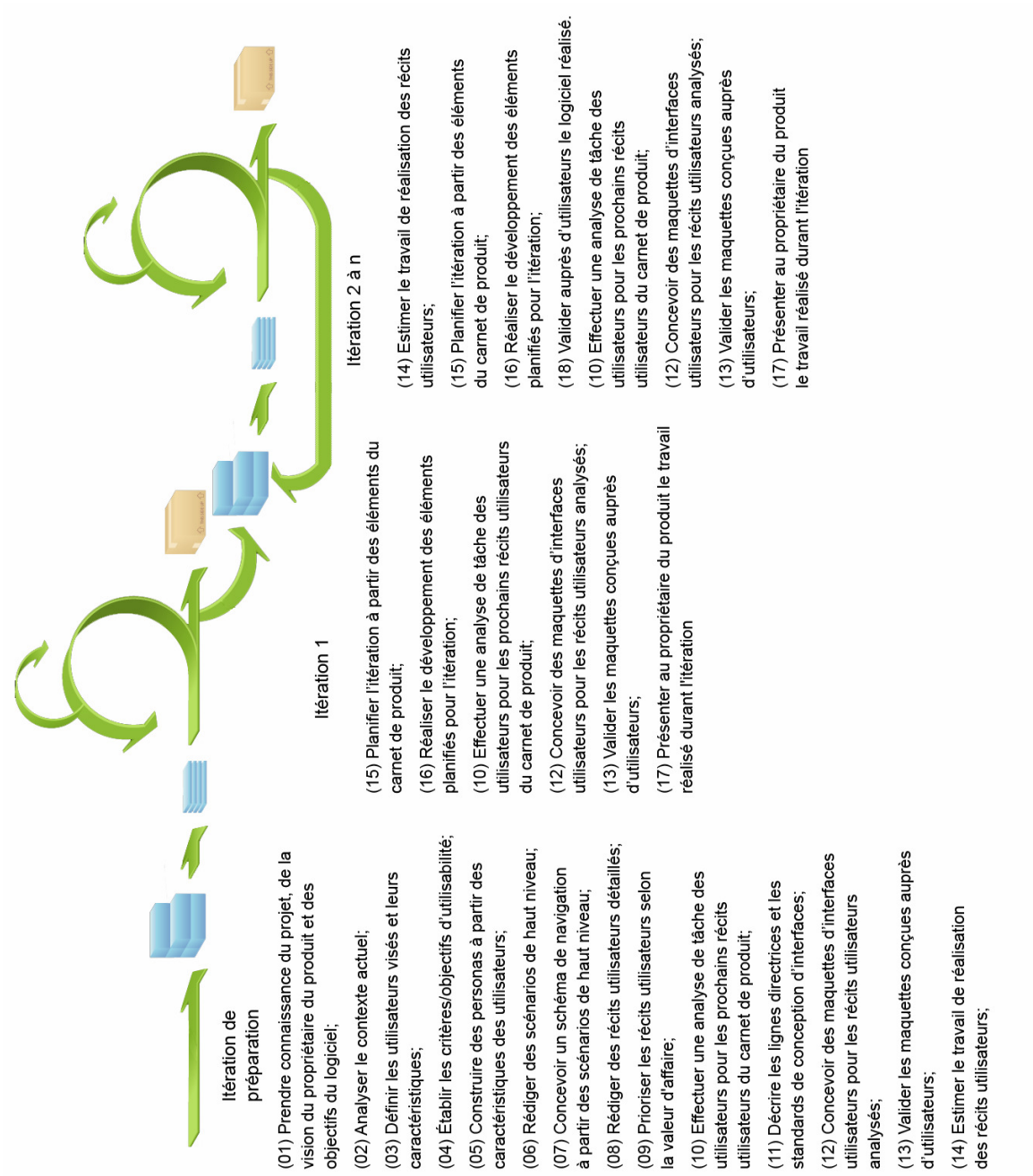
- Une description de l'activité et les conséquences de ne pas l'exécuter;
- Les intervenants recommandés;
- Le calendrier selon lequel l'activité devrait se dérouler;
- Le livrable attendu, accompagné d'une liste non exhaustive d'exemples;
- Le temps alloué pour exécuter l'activité. Le temps suggéré est basé sur une itération de 3 semaines. À moins de précision, le temps peut être ajusté de façon proportionnelle si l'itération est de durée différente;
- Une méthode allégée de réalisation permettant d'obtenir un minimum d'avantages de l'activité dans un délai raisonnable;
- Une méthode de réalisation permettant d'obtenir plus de détails et ainsi un gain plus marqué de l'intégration des activités.

Ce guide a été bâti à partir de trois sources d'information :

- L'expérience de l'auteur (depuis 2002) dans la réalisation de nombreux projets de développement logiciel selon une approche Agile/Scrum. Le guide est devenu nécessaire pour répondre aux interrogations à propos des pratiques d'ergonomie à mettre en place durant le déroulement de projets;

- Le modèle de Sy [1] qui décrit une approche de conception centrée utilisateur en mode itératif. Certains éléments de ce modèle ne sont pas entièrement en ligne avec les principes agiles. Nous tentons de corriger la situation en adhérant de plus près aux valeurs [2] et aux principes agiles [3];
- Le modèle de Mayhew [4] sur le cycle de l'ingénierie de l'utilisabilité (Usability Engineering Lifecycle). Ce cycle de conception est reconnu et utilisé par des praticiens de l'ergonomie. Il s'oriente vers des principes agiles en proposant, entre autres, une approche itérative et permettant des points d'inspection pour réviser les documents et les modèles élaborés. Comme pour le modèle de Sy, il n'est pas entièrement en ligne avec les principes agiles parce qu'il propose de réaliser un certain nombre de travaux en amont du développement du logiciel et ce, sans vraiment tenir compte des besoins changeants énoncés par le PP.

GUIDE


Itération de préparation
(01) Prendre connaissance du projet, de la vision du propriétaire du produit et des objectifs du logiciel;
(02) Analyser le contexte actuel;
(03) Définir les utilisateurs visés et leurs caractéristiques;
(04) Établir les critères/objectifs d'utilisabilité;
(05) Construire des personas à partir des caractéristiques des utilisateurs;
(06) Rédiger des scénarios de haut niveau;
(07) Concevoir un schéma de navigation à partir des scénarios de haut niveau;
(08) Rédiger des récits utilisateurs détaillés;
(09) Prioriser les récits utilisateurs selon la valeur d'affaire;
(10) Effectuer une analyse de tâche des utilisateurs pour les prochains récits utilisateurs du carnet de produit;
(11) Décrire les lignes directrices et les standards de conception d'interfaces;
(12) Concevoir des maquettes d'interfaces utilisateurs pour les récits utilisateurs analysés;
(13) Valider les maquettes conçues auprès d'utilisateurs;
(14) Estimer le travail de réalisation des récits utilisateurs;
Itération 1
(15) Planifier l'itération à partir des éléments du carnet de produit;
(16) Réaliser le développement des éléments planifiés pour l'itération;
(10) Effectuer une analyse de tâche des utilisateurs pour les prochains récits utilisateurs du carnet de produit;
(12) Concevoir des maquettes d'interfaces utilisateurs pour les récits utilisateurs analysés;
(13) Valider les maquettes conçues auprès d'utilisateurs;
(17) Présenter au propriétaire du produit le travail réalisé durant l'itération
Itération 2 à n
(14) Estimer le travail de réalisation des récits utilisateurs;
(15) Planifier l'itération à partir des éléments du carnet de produit;
(16) Réaliser le développement des éléments planifiés pour l'itération;
(18) Valider auprès d'utilisateurs le logiciel réalisé.
(10) Effectuer une analyse de tâche des utilisateurs pour les prochains récits utilisateurs du carnet de produit;
(12) Concevoir des maquettes d'interfaces utilisateurs pour les récits utilisateurs analysés;
(13) Valider les maquettes conçues auprès d'utilisateurs;
(17) Présenter au propriétaire du produit le travail réalisé durant l'itération

1. Prendre connaissance du projet, de la vision du propriétaire du produit et des objectifs du logiciel

Description

Le PP transmet à l'équipe de développement sa vision et l'objectif qu'il tente de réaliser avec le logiciel qui sera développé. Ces éléments permettent aux membres de l'équipe de prendre des décisions éclairées lors de la conception et du développement du logiciel pour produire un logiciel qui respecte les attentes premières du PP. De plus, en connaissant l'objectif du système, l'équipe peut avoir de meilleures conversations avec le PP et même le questionner sur la pertinence de certains récits utilisateurs.

Conséquences de ne pas faire l'activité

Ne pas prendre connaissances de la vision du PP et des objectifs du logiciel empêche les participants du projet d'avoir une compréhension commune des raisons sous-jacentes à la réalisation du logiciel et augmente ainsi la possibilité d'ajouter des fonctionnalités non pertinentes au logiciel. Cela peut faire augmenter inutilement les coûts (ressource, temps et budget) du projet.

Intervenants

Propriétaire du produit Équipe

Calendrier

Itération de préparation Itération 1 Itération 2 à n

Livrables

Un court document décrivant la vision, les objectifs du logiciel et toute autre information sur le projet que le PP considère pertinente au début.

Exemples

Vision :

- *Centraliser les informations de facturation et augmenter l'efficacité du processus*

Objectifs :

- *Diminuer le temps de création d'une facture*
- *Rassembler les informations de facturation et celles de la banque de clients*
- *Permettre aux clients de gérer eux-mêmes leurs factures*

Temps alloué

3 heures

Méthode allégée

Présentation par le PP et discussion ouverte entre tous les intervenants présents.

Méthode plus élaborée

La méthode allégée est considérée suffisante.

2. Analyser le contexte actuel

Description

Le contexte actuel d'utilisation représente les connaissances sur le travail tel qu'il est effectué par les utilisateurs. Au moment de l'analyse du contexte, le travail peut être effectué d'une manière entièrement différente de celle envisagée dans le logiciel qui sera développé. C'est à partir de cette analyse qu'il est possible de faire ressortir les caractéristiques du travail en cours, les systèmes utilisés, les contraintes existantes qui influent sur le déroulement du travail.

Conséquences de ne pas faire l'activité

Avant de commencer la conception et l'analyse d'un système, il est nécessaire de connaître le contexte pour pouvoir s'y référer, autant pour repérer les conditions qui contribuent positivement au travail que pour tenter d'éliminer celles qui nuisent. Sans cette analyse, il devient difficile d'avoir une vue d'ensemble sur les opportunités et les contraintes existantes. En omettant cette activité, le travail d'analyse de tâche (Activité 10) prend une plus grande importance.

Intervenants

Propriétaire du produit Équipe

Calendrier

Itération de préparation Itération 1 Itération 2 à n

Livrables

Liste et description des caractéristiques du contexte.

Exemples

- *Travail effectué seul à un poste de travail sur un système X.*
- *Compilation essentiellement manuelle, sauf pour la transmission des données entres logiciels.*

Temps alloué

3 heures

Méthode allégée

Recueil de données du contexte à partir d'entrevues avec le PP, selon sa connaissance du travail des utilisateurs.

Méthode plus élaborée

Recueil de données du contexte à partir d'entrevues avec des utilisateurs.

3. Définir les utilisateurs visés et leurs caractéristiques

Description

L'équipe travaille avec le PP pour identifier les utilisateurs visés par le logiciel. Une partie de cette information provient de la vision et des objectifs du logiciel, puisqu'en élaborant sa vision du logiciel, le PP est amené à définir le public visé. Il s'agit de décrire les utilisateurs visés de manière assez détaillée mais sans pour autant être exhaustif. Ceci facilitera la prise de décision dans la majorité des activités qui suivent.

Conséquences de ne pas faire l'activité

Sans la connaissance des caractéristiques des utilisateurs, les choix de conception ne peuvent pas être optimaux et pire, ils peuvent cibler les mauvais utilisateurs.

Intervenants

Propriétaire du produit Équipe

Calendrier

Itération de préparation Itération 1 Itération 2 à n

En amont de la rédaction de récits utilisateurs afin qu'ils ciblent bien les bons utilisateurs.

Livrables

Liste de profils-types d'utilisateurs et des principales caractéristiques du travail accompli.

Exemples

Commis à la comptabilité; assiste au suivi et à la création des factures clients et fournisseurs, connaît le domaine et le logiciel ABC qu'il utilise depuis 3 ans; niveau universitaire en administration, débrouillard et minutieux, apprend rapidement.

Temps alloué

1 jour

Méthode allégée

Discussion avec le PP ou l'équipe de mise en marché pour connaître les caractéristiques des utilisateurs visés.

Méthode plus élaborée

Entrevue avec les utilisateurs visés (par ex., 2 ou 3 par rôle) pour obtenir un portrait de leurs rôles et leurs caractéristiques.

4. Établir les critères/objectifs d'utilisabilité

Description

Les critères d'utilisabilité permettent d'établir comment sera évaluée l'utilisation du logiciel. Ils sont déterminés à partir des caractéristiques des utilisateurs (motivation, expérience à exécuter le travail, fréquence d'exécution, caractéristiques physiques, etc.) afin que les utilisateurs puissent tirer plein potentiel du logiciel. Les critères d'utilisabilité (temps d'exécution d'une tâche, fréquence des erreurs, degré d'autonomie de l'utilisateur, niveau de satisfaction, etc.) guideront la conception du logiciel tout au long du développement.

Malgré qu'il n'y ait pas encore eu d'analyse de tâches qui, normalement, facilite la définition des critères d'utilisabilité, nous proposons d'en définir dès le début du projet pour faciliter le déroulement des autres activités. Les critères énoncés durant l'itération de préparation peuvent et devraient être revus à chaque itération.

Conséquences de ne pas faire l'activité

Quoiqu'il soit possible d'évaluer les interfaces lors de tests d'utilisabilité avec des utilisateurs sans pour autant connaître les critères d'utilisabilité, les interfaces risquent de manquer de cohérence entre elles puisque la conception n'est pas guidée par des caractéristiques d'interfaces connues.

Intervenants

Propriétaire du produit Équipe

Calendrier

Itération de préparation Itération 1 Itération 2 à n

Livrables

Liste de critères d'utilisabilité et l'objectif visé par les critères.

Exemples

- *Cohérence entre les interfaces*
- *Minimiser les erreurs de saisie*
- *Faciliter le travail fréquent plus que l'apprentissage*
- *Utiliser la terminologie des utilisateurs*

Temps alloué

3 heures

Méthode allégée

Extraire les critères à partir d'entrevues avec le PP. Ce dernier amène l'équipe à connaître les critères qui vont de pair avec le retour sur investissement anticipé.

Méthode plus élaborée

Extraire les critères à partir des caractéristiques des utilisateurs et des entrevues avec le PP. Ce dernier amène l'équipe à connaître les critères qui vont de pair avec le retour sur investissement anticipé.

5. Construire des personas à partir des caractéristiques des utilisateurs

Description

Afin de rendre plus concrètes les caractéristiques des utilisateurs, il est proposé de construire des personas et de les utiliser dans la rédaction de récits utilisateurs. On s'assure ainsi que l'équipe de développement vise bien les utilisateurs ciblés par le logiciel.

Conséquences de ne pas faire l'activité

La rédaction des récits utilisateurs perd sa spécificité puisqu'elle ne cible pas un utilisateur précis.

Intervenants

Propriétaire du produit Équipe

Calendrier

Itération de préparation Itération 1 Itération 2 à n

En amont de la rédaction de récits utilisateurs

Livrables

Un ensemble de personas (environ 5) identifiant des profils-types d'utilisateurs et des caractéristiques précises qui aident à prendre des décisions pour la création des récits et la conception.

Exemples

Paul, commis, a 28 ans et travaille chez ACME depuis 5 ans, il s'agit de son deuxième emploi dans le domaine de la comptabilité. Comme il n'a pas encore son titre comptable, il assiste les comptables dans le suivi quotidien de la facturation. Il apprend rapidement les nouveaux concepts et connaît bien les systèmes informatiques, tant qu'il ne s'agit pas de programmer. Malgré sa capacité d'apprentissage rapide, il est peu patient avec les systèmes difficiles à utiliser. Lorsqu'il rencontre de tels systèmes, il préfère travailler de manière manuelle, même si c'est moins efficace.

Temps alloué

1 jour

Méthode allégée

À partir des caractéristiques des utilisateurs, extraire les points communs en termes de comportements, d'attributs et d'attentes et les regrouper pour former des « personnes » types.

Méthode plus élaborée

La méthode allégée est considérée comme suffisante.

6. Rédiger des scénarios de haut niveau

Description

À partir des objectifs du système, définir des scénarios de haut niveau (pour le système). Ceux-ci sont similaires à des thèmes ou des « modules » à inclure dans le logiciel; par exemple, un logiciel de facturation est découpé en modules de clients, de factures, d'utilisateurs du logiciel, etc. Les scénarios de haut niveau n'ont pas à être exhaustifs durant l'itération de préparation puisque de nouveaux scénarios seront rédigés et détaillés durant les itérations à venir. Il suffit d'en rédiger suffisamment pour représenter ce que nous connaissons au moment de l'itération de préparation du système.

Conséquences de ne pas faire l'activité

Les scénarios de haut niveau permettent à tous les intervenants de comprendre comment se découpe le système. Sans ceux-ci, il devient plus difficile d'avoir une vue d'ensemble de la portée du logiciel et de prendre des décisions judicieuses lors de la conception.

Intervenants

Propriétaire du produit Équipe

Calendrier

Itération de préparation Itération 1 Itération 2 à n

Livrables

Liste de scénarios de haut niveau priorisés selon leur valeur d'affaire par le PP.

Exemples

- *Créer une facture à partir des informations d'un client.*
- *Créer un client à partir d'une nouvelle facture.*
- *(Le client) Recherche toutes factures émises pour un fournisseur.*

Temps alloué

3 heures

Méthode allégée

Rédiger des scénarios de haut niveau à partir des explications fournies par le PP à l'équipe de développement.

Méthode plus élaborée

Exécuter l'activité simultanément avec l'activité 7, utiliser la méthode de « User story mapping ».
(http://www.agileproductdesign.com/presentations/user_story_mapping/index.html)

7. Concevoir un schéma de navigation à partir des scénarios de haut niveau

Description

Concevoir un schéma permettant de voir et de comprendre comment les différents scénarios de haut niveau sont interreliés. Ceci donne à toute l'équipe une vision sur les différents modules et sur les liens potentiels entre eux ou permet de clarifier leur indépendance. Dans cette activité, le schéma de navigation ne traite pas spécifiquement des différents menus qui seront présents dans les interfaces utilisateurs, ni des détails de la séquence des interfaces. Il s'agit bien de comprendre les relations entre les modules présentés dans les scénarios de haut niveau.

Conséquences de ne pas faire l'activité

Le logiciel risque de devenir incohérent dans son ensemble s'il n'est pas possible de rattacher les éléments à des modules et d'avoir une idée de la manière dont les modules se rattachent les uns aux autres.

Intervenants

Propriétaire du produit Équipe

Calendrier

Itération de préparation Itération 1 Itération 2 à n

Livrables

Schéma de navigation entre les modules.

Exemples

Temps alloué

3 heures

Méthode allégée

Une méthode du *card sorting* où chaque scénario de haut niveau est écrit sur une carte, et par la suite le PP et les membres de l'équipe trient et organisent les cartes pour faire ressortir les liens existants.

Méthode plus élaborée

Exécuter l'activité simultanément avec l'activité 6, utiliser la méthode de « User story mapping »
(http://www.agileproductdesign.com/presentations/user_story_mapping/index.html)

8. Rédiger des récits utilisateurs détaillés

Description

Les récits utilisateurs sont rédigés à partir des scénarios de haut niveau et seront les éléments qui seront inclus dans le travail d'une itération.

Conséquences de ne pas faire l'activité

Concevoir un logiciel sans rédiger des récits utilisateurs risque de faire ressortir des besoins qui ne sont pas réels ou qui ne visent pas les bons utilisateurs.

Intervenants

Propriétaire du produit Équipe

Calendrier

Itération de préparation Itération 1 Itération 2 à n

Livrables

Un ensemble de récits utilisateurs qui détaillent les scénarios de haut niveau rédigés précédemment.

Exemples

- *En tant que commis, lorsque je crée une facture suite à l'achat d'un nouveau client, afin d'éviter les erreurs de ressaisie, je veux que les informations du nouveau client soient automatiquement transférées dans le système « Client ».*
- *En tant que commis, lorsque je crée une facture suite à l'achat par un client, afin d'avoir une facture complète, je veux que les informations de contact soient automatiquement inscrites sur la facture.*
- *En tant que client, lorsque je suis connecté au système, afin d'avoir une facture personnalisée, je veux que mes informations de contact soient automatiquement inscrites sur ma facture.*

Temps alloué

3 heures

Méthode allégée

La rédaction de récits débute par une conversation entre le PP et les membres de l'équipe pour mettre en valeur un besoin. La conversation fait ressortir l'utilisateur visé, l'objectif qu'il tente d'atteindre et l'action à exécuter.

Méthode plus élaborée

La méthode allégée est considérée comme suffisante.

9. Prioriser les récits utilisateurs selon la valeur d'affaire

Description

Le PP ordonne les récits utilisateurs en fonction de leur valeur d'affaire et de leur l'utilité pour les utilisateurs. Les membres de l'équipe travaillent de pair avec le PP pour qu'il considère les récits qui ajoutent de l'utilité au logiciel pour l'utilisateur, même s'ils semblent présenter une plus petite valeur d'affaire.

Conséquences de ne pas faire l'activité

Un carnet de produit non priorisé ne permet pas de livrer un logiciel qui a de la valeur à chaque itération puisque les fonctionnalités développées ne sont pas ordonnées.

Intervenants

Propriétaire du produit Équipe

Calendrier

Itération de préparation Itération 1 Itération 2 à n

Livrable

Liste ordonnée de récits utilisateurs. Cette liste fait partie du carnet de produit.

Exemples

- *En tant que client, lorsque je suis connecté au système, afin d'avoir une facture personnalisée, je veux que mes informations de contact soient automatiquement inscrites sur ma facture.*
- *En tant que commis, lorsque je crée une facture suite à l'achat par un client, afin d'avoir une facture complète, je veux que les informations de contact soient automatiquement inscrites sur la facture.*
- *En tant que commis, lorsque je crée une facture suite à l'achat d'un nouveau client, afin d'éviter les erreurs de ressaisie, je veux que les informations du nouveau client soient automatiquement transférées dans le système « Client ».*

Temps alloué

3h

Méthode allégée

Les récits utilisateurs sont généralement inscrits sur des cartes index. Le PP peut alors ordonner les récits en triant les cartes selon la valeur d'affaire qu'il associe à chaque récit. L'équipe de développement peut aider le PP à ordonner les récits en l'amenant à considérer des contraintes techniques (par ex., un récit qui ne peut être réalisé avant qu'une intégration avec un système externe ne soit complétée), des informations relatives à la tâche de l'utilisateur (par ex., des contraintes d'environnement qui ne sont pas présentes dans les récits abordés précédemment), etc.

Méthode plus élaborée

La méthode allégée est considérée comme suffisante.

10. Effectuer une analyse de tâche des utilisateurs pour les prochains récits utilisateurs du carnet de produits

Description

Une analyse de tâche permet de décrire et de comprendre le travail réel de l'utilisateur et de définir le modèle de la tâche. Il est probable que le système impose une nouvelle manière de réaliser le travail. L'analyse de la tâche actuelle demeure utile pour prendre en considération toutes les informations nécessaires à l'utilisateur. Un récit utilisateur peut comprendre plusieurs tâches à effectuer qui correspondent aux différents moments de l'activité de l'utilisateur et qui déterminent en grande partie les comportements de l'utilisateur.

Dans ce guide, l'analyse de tâche est réalisée après la définition des récits utilisateurs et non au début comme on s'y attend dans une approche de CCU. Ce choix est volontairement fait pour réduire la portée de l'analyse de tâche à ce que le PP tente de réaliser dans le logiciel. Ainsi, il est possible de réaliser l'analyse de tâche concernant le récit traité dans une itération et en faire ressortir les informations utiles et nécessaires à la conception et au développement du récit correspondant.

Conséquences de ne pas faire l'activité

Sans l'analyse de tâche de l'utilisateur, il est impossible de connaître certaines informations sur la tâche. Les seules discussions avec le PP risquent d'être insuffisantes car ce dernier ne connaît pas nécessairement toutes les tâches de l'utilisateur, ni les raccourcis qu'il utilise quotidiennement.

Intervenants

Propriétaire du produit Équipe

Calendrier

Itération de préparation Itération 1 Itération 2 à n

L'analyse de tâche est réalisée durant l'itération qui précède le développement.

Livrables

Modèle de tâches des utilisateurs pour le récit envisagé.

Exemples

- *Le client accède au système pour faire un achat.*
- *Recherche, trouve et achète les produits qu'il désire.*
- *Ouvre un fichier contenant les informations de facturation.*
- *Copie les informations à partir du fichier vers la facture en ligne.*
- *Complète l'achat.*

Temps alloué

1 heure par récit utilisateur

Méthode allégée

Le recueil de données pour l'analyse de tâche peut se faire par entrevue avec le PP, selon sa connaissance du travail des utilisateurs.

Méthode plus élaborée

Le recueil de données pour l'analyse de tâche peut se faire par entrevue des utilisateurs et sur la base d'observations des utilisateurs au travail.

11. Décrire les lignes directrices et les standards de conception d'interfaces

Description

Les lignes directrices de conception d'interfaces sont de précieuses références pour l'équipe Scrum. Elles portent sur tout un ensemble d'éléments de l'interface (ex., terminologie, navigation, guidage, disposition graphique, erreurs, couleurs, etc.) et décrivent le comportement des diverses composantes de l'interface (ex., menus, fenêtres, retour d'information, messages, etc.). Ces éléments débutent parfois par des contraintes ou des lignes directrices énoncées par l'organisation.

Conséquences de ne pas faire l'activité

Les interfaces développées pour le système risquent de manquer de cohérence entre elles ou avec celles des autres systèmes de l'organisation. Cela peut avoir comme effet d'allonger la période d'apprentissage pour les utilisateurs.

Intervenants

Propriétaire du produit Équipe

Calendrier

Itération de préparation Itération 1 Itération 2 à n

Une version préliminaire est élaborée durant l'itération de préparation. Les règles évoluent tout au long du projet et sont revues au cours des itérations.

Livrables

Un guide de règles de conception, enrichi d'exemples.

Exemples

- *Tous les messages à l'utilisateur sont affichés dans une fenêtre « modale ».*
- *Dans les formulaires, les libellés sont affichés au dessus du champ correspondant.*
- *Les éléments accessibles par la souris sont aussi accessibles par raccourcis clavier.*
- *Les boutons permettent à l'utilisateur d'effectuer une action, les hyperliens (soulignés et en bleu) permettent la navigation dans le système.*

Temps alloué

3 heures

Méthode allégée

Faire ressortir les lignes directrices de l'organisation et les inscrire dans un guide de règles. Selon le domaine d'application du logiciel en réalisation, il existe des lignes directrices qui sont partagées publiquement, ces références sont un point de départ intéressant pour tout projet.

Méthode plus élaborée

La méthode allégée est considérée comme suffisante.

12. Concevoir des maquettes d'interfaces utilisateurs pour les récits utilisateurs analysés

Description

La conception de maquettes devrait être faite par tous les membres de l'équipe Scrum. L'ergonome de l'équipe s'assure continuellement de la prise en compte des caractéristiques des utilisateurs. Le développeur s'assure que les concepts sont techniquement réalisables, en considérant les capacités et les limites des technologies utilisées dans la réalisation du logiciel

Conséquences de ne pas faire l'activité

Les maquettes d'interfaces utilisateurs sont un moyen efficace de montrer le comportement attendu d'une interface. Sans la représentation visuelle par une maquette, un récit risque de ne pas être interprété correctement par les membres de l'équipe. La maquette réduit les risques de mauvaises interprétations du contenu des récits.

Intervenants

Propriétaire du produit Équipe

Calendrier

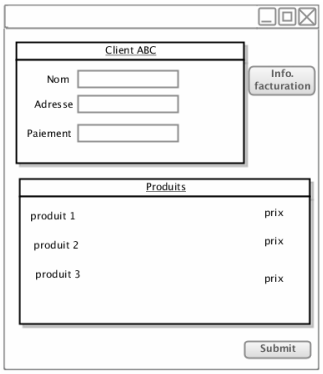
Itération de préparation Itération 1 Itération 2 à n

La conception de maquettes est réalisée durant l'itération précédant la réalisation par les développeurs.

Livrables

Maquettes d'interfaces utilisateurs, accompagnées du récit utilisateur représenté.

Exemples



Temps alloué

2 heures par maquette

Méthode allégée

La conception de maquettes par la méthode de prototypage papier est simple et rapide. Cette méthode est utile dans le cadre d'une itération Scrum puisqu'elle livre une grande valeur, sans utiliser trop de ressources de l'équipe et permet d'effectuer plusieurs modifications rapides au sein de l'itération Scrum.

Méthode plus élaborée

La méthode allégée est considérée comme suffisante.

13. Valider les maquettes conçues, auprès d'utilisateurs

Description

La validation de maquettes d'interface fait ressortir des corrections ou des améliorations à faire. Celles-ci peuvent être rapidement apportées aux maquettes, avant même le début du développement du récit.

Conséquences de ne pas faire l'activité

Effectuer plus tard ou ne pas effectuer de validation des maquettes fait augmenter le coût de correction des anomalies. Les coûts en efforts et en temps de correction sont minimes à cette étape en comparaison de ceux encourus une fois le développement débuté. De plus, puisque les répercussions sur l'ensemble du logiciel deviennent plus grandes, le coût de correction continue d'augmenter au fur et à mesure que les incréments de logiciel sont ajoutés.

Intervenants

Propriétaire du produit Équipe

Calendrier

Itération de préparation Itération 1 Itération 2 à n

Livrables

Liste de recommandations d'améliorations et corrections suggérées pour chaque maquette.

Exemples

Lorsque la facture s'affiche, le client cherche ses informations; il s'attend à les voir automatiquement apparaître; il ne voit pas immédiatement le bouton « info. facturation ». Lorsqu'il le voit, il se demande pourquoi le système ne le fait pas par lui-même.

Correction suggérée : Lorsque la fenêtre se charge, insérer les informations du client à l'endroit approprié.

Temps alloué

3 à 5 utilisateurs par maquette et 30 minutes par utilisateur.

Méthode allégée

Une méthode similaire aux tests d'utilisabilité où on demande aux participants d'exécuter, avec les maquettes, un certain nombre de tâches qui correspondant aux objectifs des récits utilisateurs sélectionnés.

Méthode plus élaborée

La méthode allégée est considérée comme suffisante.

14.Estimer le travail de réalisation des récits utilisateurs

Description

L'estimation du travail de réalisation des récits utilisateurs par l'équipe permet au PP de connaître le coût de développement des récits qu'il rédige. Elle est un pré-requis à la planification des itérations puisqu'elle permet de connaître avec précision la quantité de travail réalisable durant cette itération.

Conséquences de ne pas faire l'activité

Un travail dont les efforts requis pour le faire et la durée n'ont pas été estimés ne peut être planifié adéquatement.

Intervenants

Propriétaire du produit	Équipe

Calendrier

Itération de préparation	Itération 1	Itération 2 à n

L'estimation a lieu avant la séance de planification de l'itération.

Livrables

Une estimation de l'effort de réalisation associée aux récits utilisateurs du carnet de produit.

Exemples

N/A

Temps alloué

15 à 30 minutes par récit à estimer.

Méthode allégée

La technique du Planning Poker® (marque déposée de Mountain Goat Software, LLC) est simple et efficace pour arriver à une estimation juste.

Méthode plus élaborée

La méthode allégée est considérée comme suffisante.

15.Planifier l'itération à partir des éléments du carnet de produit

Description

Durant la séance de planification d'itération, l'équipe s'engage avec le PP à réaliser une certaine quantité de travail durant l'itération qui débute. Le travail à réaliser concerne autant la conception de récits utilisateurs par les ergonomes que le développement par les développeurs des récits préalablement conçus par l'équipe.

Conséquences de ne pas faire l'activité

Comme dans tout projet de développement logiciel, la planification est nécessaire, même pour un projet se déroulant dans un mode agile. Par contre, la planification se limite ici au contenu de l'itération qui débute.

Intervenants

Propriétaire du produit Équipe

Calendrier

Itération de préparation Itération 1 Itération 2 à n

La planification est la première activité réalisée dans l'itération.

Livrables

Le carnet d'itération contenant les récits utilisateurs qui seront conçus ou développés par l'équipe.

Exemples

N/A

Temps alloué

3 heures

Méthode allégée

Une discussion entre les intervenants du projet pour planifier la réalisation de récits qui livrent de la valeur au PP, tout en s'assurant que le travail est bien complété à la fin de l'itération, le tout en respectant les critères de qualité décidés par l'équipe.

Méthode plus élaborée

La méthode allégée est considérée comme suffisante.

16.Réaliser le développement des éléments planifiés pour l'itération

Description

À cette étape, l'équipe travaille sur la conception et la réalisation technique des récits utilisateurs qui ont été planifiés lors de la séance de planification de l'itération courante.

Conséquences de ne pas faire l'activité

Cette étape ne peut pas être omise puisqu'elle est essentielle dans tout projet de développement logiciel. Sans réalisation, le logiciel demeure à l'étape de concept et n'apporte pas de valeur au PP.

Intervenants

Propriétaire du produit	Équipe

Calendrier

Itération de préparation	Itération 1	Itération 2 à n

Livrables

Un incrément de logiciel fonctionnel qui respecte les critères de qualité de l'équipe.

Exemples

N/A

Temps alloué

La durée d'une l'itération est utilisée pour la réalisation du développement.

Méthode allégée

L'équipe utilise la méthode de conception et de programmation qu'elle considère comme la plus appropriée pour la réalisation du logiciel.

Méthode plus élaborée

La méthode allégée est considérée comme suffisante.

17. Présenter au propriétaire du produit le travail réalisé durant l'itération

Description

À la fin de chaque itération, l'équipe présente au PP le travail réalisé durant l'itération. La présentation au PP est la confirmation par l'équipe que le travail réalisé durant l'itération correspond effectivement aux attentes énoncées dans les récits utilisateurs et à l'engagement pris durant la séance de planification.

Cette présentation peut être considérée comme un test d'utilisabilité préliminaire où le PP agit en tant qu'utilisateur. C'est lors de l'itération suivante que des utilisateurs supplémentaires seront amenés à faire des validations. Les comportements du PP ne doivent pas être considérés comme représentatifs du comportement typique d'un utilisateur du système. Par contre, ils permettent d'obtenir un premier ensemble de données.

Conséquences de ne pas faire l'activité

Le PP ne s'assure pas que le logiciel qui est développé correspond bien à ce qu'il s'attend et que l'équipe crée un logiciel de valeur pour lui.

Intervenants

Propriétaire du produit Équipe

Calendrier

Itération de préparation Itération 1 Itération 2 à n

Livrables

Un principe agile mentionne qu'un logiciel fonctionnel est la mesure principale de l'avancement. Alors, l'incrément de logiciel réalisé par l'équipe de développement est présenté au PP. De plus, les ergonomes présentent les maquettes conçues et les résultats de validations faites auprès des utilisateurs afin de permettre au PP de connaître les modifications à apporter au logiciel réalisé.

Exemples

N/A

Temps alloué

3 heures

Méthode allégée

Les récits réalisés sont présentés au PP et ce dernier tente de les exécuter avec le logiciel. Ceci est en sorte un premier test d'utilisabilité fait dans l'itération de réalisation.

Méthode plus élaborée

La méthode allégée est considérée comme suffisante.

18. Valider le logiciel réalisé auprès d'utilisateurs

Description

Avant la mise en production du logiciel, la validation auprès d'utilisateur est un moyen très efficace pour évaluer ce logiciel et y apporter des modifications, si nécessaire. La validation est effectuée durant une séance de tests d'utilisabilité où l'équipe observe des utilisateurs face au logiciel. Cette méthode permet de relever les problèmes que rencontre l'utilisateur avec le logiciel.

Les tests d'utilisabilité permettent de connaître le comportement des utilisateurs et de vérifier qu'il correspond à celui attendu. Lorsque nécessaire, l'équipe suggère des modifications au mode d'utilisation du logiciel afin qu'il corresponde mieux aux attentes des utilisateurs.

La validation par tests d'utilisabilité à chaque itération peut représenter beaucoup de travail, mais les tests sont effectués sur la partie du logiciel qui concerne le récit utilisateur qui a été développé, et non pas sur un scénario complet.

Les corrections/modifications qui sont faites créent de nouvelles cartes dans le carnet de produit et le PP doit les prioriser, de la même manière qu'il le fait avec les récits utilisateurs.

Conséquences de ne pas faire l'activité

Le PP ne reçoit pas de rétroactions sur l'utilisation réelle du logiciel avant sa mise en production. Cela engendre des coûts plus élevés de correction.

Intervenants

Propriétaire du produit Équipe

Calendrier

Itération de préparation Itération 1 Itération 2 à n

Les tests d'utilisabilité sont exécutés durant l'itération qui suit le développement.

Livrables

Liste de recommandations d'améliorations et corrections suggérées pour les interfaces utilisateurs.

Exemples

- *Lorsque la facture s'affiche, le client cherche ses informations; il s'attend à les voir automatiquement apparaître; il ne voit pas immédiatement le bouton « Insérer info. facturation ». Lorsqu'il le voit, il demande pourquoi le système ne le fait pas par lui-même.*
    - *Correction suggérée : Lorsque la fenêtre se charge, insérer les informations du client à l'endroit approprié.*

Temps alloué

La validation des interfaces devrait être faite auprès d'un minimum de 3 à 5 utilisateurs et durer moins de 30 minutes pour chacun.

Méthode allégée

Durant les tests d'utilisabilité, on demande aux utilisateurs d'exécuter un certain nombre de tâches correspondant aux objectifs des récits utilisateurs sélectionnés afin d'observer si le comportement du logiciel répond à leurs attentes et besoins.

Méthode plus élaborée

La méthode allégée est considérée comme suffisante.

RÉFÉRENCES

[1] D. Sy, « Adapting Usability Investigations for Agile User-Centered Design », Journal of Usability Studies, Vol. 2, No 3, pp. 112-132, 2007

[2] K. Beck, M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R.C. Martin, S. Mellor, K. Schwaber, J. Sutherland, D. Thomas « Manifesto for Agile Software Development ». Manifesto for Agile Software Development. [En ligne]. Disponible: http://www.agilemanifesto.org/. [Consulté le 23 juillet 2009]

[3] K. Beck, M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R.C. Martin, S. Mellor, K. Schwaber, J. Sutherland, D. Thomas « Principles behind the Agile Manifesto ». Principles behind the Agile Manifesto. [En ligne]. Disponible: http://www.agilemanifesto.org/principles.html [Consulté le 23 juillet 2009]

[4] D.J. Mayhew, The Usability Engineering Lifecycle: A Practitioner's Handbook for User Interface Design, 1re édition, Morgan Kaufmann, 1999.

ANNEXE 2 – Certificat d'acceptation d'un projet de recherche avec des sujets humains

ÉCOLE
POLYTECHNIQUE
MONTRÉAL

CERTIFICAT D'ACCEPTATION D'UN PROJET DE RECHERCHE PAR LE COMITÉ D'ÉTHIQUE DE LA RECHERCHE AVEC DES SUJETS HUMAINS DE L'ÉCOLE POLYTECHNIQUE

Comité d'éthique de la recherche avec des sujets humains

Adresse civique :
Campus de l'Université de Montréal
900, boul. Édouard-Montpetit
École Polytechnique
2500, chemin de Polytechnique
H3T 1J4

Adresse postale :
C.P. 6079, succursale Centre-ville
Montréal (Québec) Canada
H3C 3A7

Téléphone : (514) 340-4990
Télécopieur : (514) 340-4992

École affiliée à l'Université de Montréal

Montréal, le 22 juin 2010

M. Jean-François Proulx
M. Jean-Marc Robert
Département de mathématiques et génie industriel
École Polytechnique de Montréal

N/Réf : Dossier CÉR-09/10-13

Messieurs,

J'ai le plaisir de vous informer que les membres du Comité d'éthique de la recherche restreint ont pris connaissance des précisions apportées à leurs questions ainsi que des documents rajoutés à votre dossier intitulé « *Intégrer une approche de conception centrée utilisateur à une approche agile de développement logiciel* ».

Les membres du Comité ont recommandé l'approbation de ce projet sur la base des documents envoyés par courriel à Mme Roehrig en date du 16 juin 2010.

Veuillez noter que le présent certificat est valable pour le projet tel que soumis au Comité d'éthique de la recherche avec des sujets humains. La secrétaire du Comité d'éthique de la recherche avec des sujets humains devra immédiatement être informée de toute modification qui pourrait être apportée ultérieurement au protocole expérimental, de même que de tout problème imprévu pouvant avoir une incidence sur la santé et la sécurité des personnes impliquées dans le projet de recherche (sujets, professionnels de recherche ou chercheurs).

Nous vous prions également de nous faire parvenir un bref **rapport annuel** ainsi qu'un avis à la fin de vos travaux.

Je vous souhaite bonne chance dans vos travaux de recherche,

Bernard Lapierre, président
Comité d'éthique de la recherche avec des sujets humains

c.c. : Céline Roehrig, DRI